UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA TÉCNICA

INFORMÁTICA DE GESTIÓN

PROYECTO FIN DE CARRERA

Sistema de Navegación Local en Entornos Urbanos para un Vehículo Autónomo.

Autor: Juan José Rodríguez Castaño

Tutor: Raúl Arrabales Moreno

NOVIEMBRE 2009

ÍNDICE

ÍNDICE DE FIGURAS

AGRADECIMIENTOS

A mis padres y mis amigos por el apoyo ofrecido en toda la carrera, en especial, en el primer año.

A mi tutor por su paciencia y sus acertados consejos que me guiaron en todo momento.

Y a esa persona que estuvo conmigo durante casi toda la carrera y que ahora nuestros caminos se han separado

1.- INTRODUCCIÓN

El proyecto que se presenta a continuación propone un sistema de control automático de un vehículo sin necesidad de conductor. El vehículo incorpora una serie de sensores como son el láser para la detección de objetos y obstáculos, cámaras integradas para ver la carretera, GPS integrado sin el mapa de la ciudad donde el vehículo va a circular y sensores en los límites exteriores del coche que indican si el vehículo ha impactado con algún objeto. Este sistema de control debe ser eficiente además de rápido. En definitiva debe cumplir con todos los requisitos que hacen útil y usable un software.

Realizar este proyecto sería altamente costoso si se tuviera que realizar con elementos reales. Por suerte, para la realización del mismo se usará un entorno simulado. Una ciudad pequeña donde se sitúa el vehículo en un punto cualquiera y donde debe rodar sin colisionar con los objetos que se presenten en su camino o con otros vehículos que circulan por el mismo entorno y además circulando de manera correcta y segura.

OBJETIVOS

Los objetivos del proyecto son:

- Diseñar e implementar un sistema de control automático de un vehículo sin que ningún humano interactúe con él, más concretamente, un robot autónomo móvil, aunque para este proyecto se utilice la simulación de un vehículo real.

- Evitar impactos y colisiones con otros vehículos que se encuentran en el entorno simulado o con objetos estáticos (semáforos, farolas, edificios, etc.).

- Llevar el vehículo por la calle de manera efectiva y tal y como lo haría un humano, es decir, el programa final debe conducir el coche, llevándolo por su carril sin interferir con los demás usuarios (vehículos).

- Respetar las normas de tráfico existentes. En el mapa del proyecto solo existen semáforos y entradas a rotondas, pero se intentará que el vehículo en su movimiento cumpla con las normas de tráfico más comunes.

- Control automático del vehículo usando técnicas de visión artificial (empleando las cámaras con las que está equipado el vehículo simulado) para el analizar las imágenes capturadas por las cámaras y tomar decisiones en función de la posición del vehículo respecto a la carretera, teniendo en cuenta todos los elementos (carriles, semáforos, curvas, etc.).

- Realizar un sistema complejo que tome decisiones correctamente manteniendo un margen de error muy pequeño.

Para la consecución de estos objetivos, se han de realizar una serie de investigaciones previas aparte del desarrollo del sistema de control que se desea implementar. Todo ello se explicará en este documento cuyo objetivo es reflejar el trabajo realizado, tanto a la hora de realizar el sistema de control como la investigación previa que se ha realizado.

Primeramente, al ser un sistema autónomo, se ha investigado acerca de la robótica, la rama que se encarga de todo lo relacionado con los robots. Se han tratado varios temas, la robótica de forma general, qué son los robots y posteriormente las arquitecturas robóticas que son la base para la realización de un sistema robótico autónomo.

Una vez obtenidos estos conocimientos, se analiza de manera más o menos minuciosa el vehículo que utilizaremos en este proyecto, que, aunque tenga forma de automóvil real, no es más que un robot autónomo

móvil. Por ello, se ha estudiado de qué se compone el robot autónomo móvil, para qué se idearon este tipo de robots y sus posibles aplicaciones en el mundo actual y en un futuro.

Como se ha comentado en los objetivos del proyecto, se realizará un sistema de visión artificial. Para la realización de este sistema se hará uso de las cámaras que el vehículo lleva instaladas, siempre de manera virtual. Con estas cámaras obtendremos imágenes de las cuales, con un tratamiento previo, obtendremos la información deseada para que el vehículo circule de manera correcta. Algunas de estas fuentes de información son los semáforos, obstáculos y posibles vehículos que encontrará, pero, sin duda alguna, la aplicación más grande de este sistema de visión se realizará en la localización de los carriles por los que el vehículo deberá circular. El principal objetivo es reconocer estos carriles de manera que indique al sistema si el vehículo circula correctamente o se debe realizar algún tipo de corrección en su movimiento (curvas, giros, etc.) mediante el reconocimiento de las marcas vales horizontales, es decir, las líneas que delimitan los carriles.

Todo este proyecto se realizara con Microsoft Robotics Developer Studio (MRDS), herramienta que proporciona grandes ventajas a la hora de realizar proyectos de este calibre ya que posee características que ayudan al programador a realizar su trabajo, características enfocadas a la robótica y robots autónomos, algo que se agradece enormemente, ya que poder utilizar una herramienta que se centra en el ámbito de tu trabajo es un lujo que pocos proyectos tienen al alcance. Para interactuar con las funciones de simulación de robótica que ofrece se utilizará la herramienta Microsoft Visual Studio (MSVS) que nos ofrece un entorno de programación perfectamente adaptado a las nuevas funcionalidades que ofrece MRDS. En este entorno de programación, se utilizará el lenguaje Microsoft C# 2.0 (C#) el cual permite aprovechar al máximo el MRDS. Además de estas herramientas se usaran distintas librerías obtenidas de particulares que se describirán más adelante y que serán de gran utilidad para el desarrollo del sistema.

Naturalmente, se está ante un sistema que no posee el cien por cien de eficacia debido a la complejidad del mismo y a las numerosas variables que entran en juego en la realización de este proyecto. De hecho, si miramos el mundo real de un conductor, podemos ver que incluso, nosotros mismos, los conductores, cometemos errores de medición de distancias, radio de giro, etc. Por ello se considera aceptable no tener un cien por cien de eficacia aunque, por supuesto, el objetivo del proyecto es aumentar ese porcentaje al máximo para que el recorrido del vehículo sea exitoso en la gran mayoría de casos y situaciones.

Por último, es necesario apuntar que el proyecto se va a realizar íntegramente sobre la plataforma de Robochamps Urban Challenge (RUC) que nos ofrece una base sobre la que poder trabajar cómodamente y dedicar todos los esfuerzos a realizar en lo realmente importante que es la codificación del sistema de control automático.

1.1.- ROBOCHAMPS URBAN CHALLENGE

Robochamps se trata de una competición de robótica a nivel mundial en el que se propone la construcción de sistemas robóticos en un entorno virtual que realicen unos ciertos objetivos. Esta competición es apta tanto para expertos como para programadores jóvenes con poca experiencia que quieran demostrar su potencial en el campo de la robótica. Este concurso toma como base la tecnología .NET siempre aplicada a la robótica mediante la utilización de Microsoft Robotics Developer Studio (MRDS).

Para poder participar, como se ha comentado antes, tan solo es necesario programar, Robochamps, para cada una de sus distintos concursos, ofrece un paquete básico que proporciona una estructura sobre la cual trabajar. Se ofrecen varios desafíos, entre los cuales el que se ha elegido, como marco para este proyecto es el concurso Urban Challenge.

En el soporte ofrecido por Robochamps para este concurso, se incluye un paquete de iniciación el cual contiene un entorno virtual donde se encuentra modelada una ciudad de ejemplo con el vehículo todo ello en tres dimensiones y los elementos que se necesitan para el funcionamiento autónomo del vehículo, es decir, elementos como las cámaras web, sensores de distancia, etc. Además de ello también se ofrece un interfaz de

control de vehículo, siendo capaces de controlar el vehículo de manera manual, haciendo girar el vehículo, acelerar, frenar mediante el control del interfaz sin programar una sola línea. Por lo tanto, RUC busca que los concursantes se centren en el control de robots y no en otros elementos que poco tienen que ver con la robótica y que conforman una base necesaria.

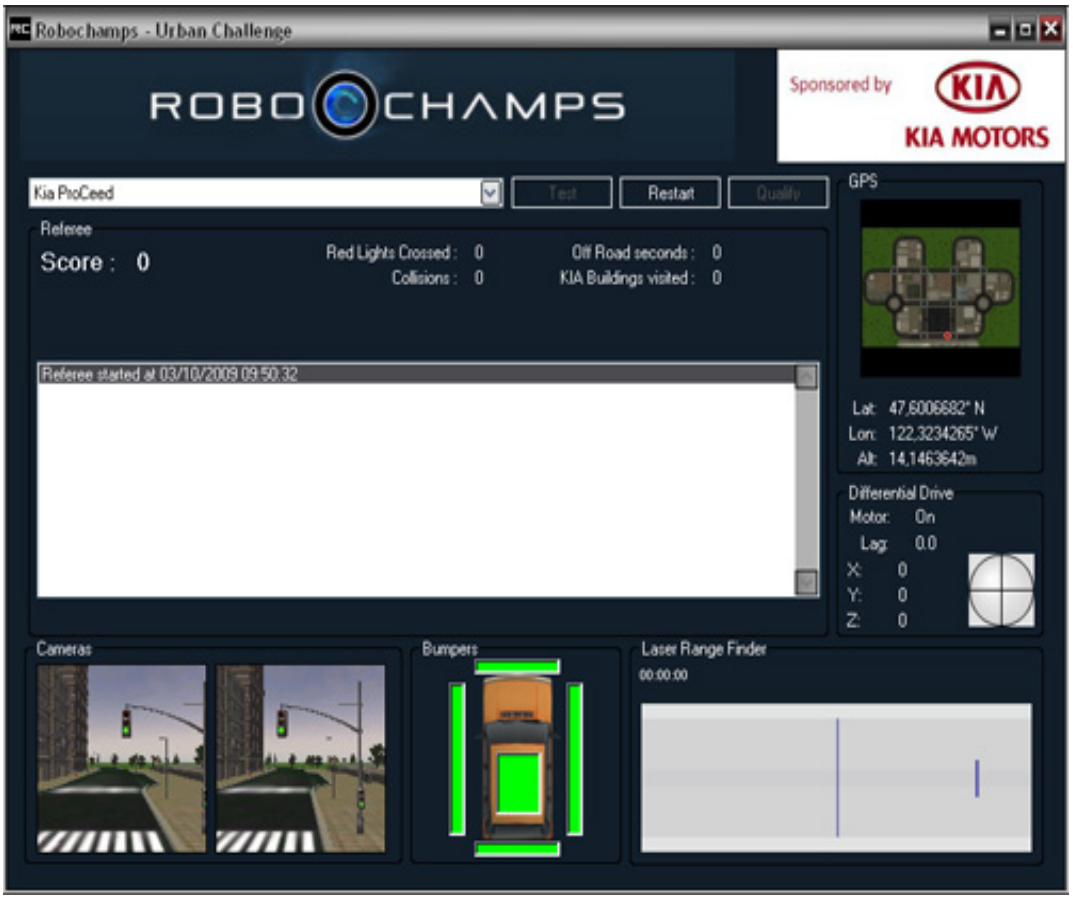


Figura 1: interfaz de control del vehículo propuesta por RUC

Además, este paquete de iniciación está escrito en el lenguaje C# con lo que se invita a realizar el concurso en dicho lenguaje.

Por otro lado, los componentes virtuales que simulan los componentes hardware necesarios para el funcionamiento del vehículo son:

- Motor: se proporciona un servicio que simula un motor estándar de un solo eje y sin dirección. Para realizar los giros es necesario aplicar distintas potencias a cada una de las ruedas por separado, algo que en los vehículos corrientes no es así, ya que poseen un eje directriz. Por ello se ha simulado el giro de las ruedas en RUC de manera visual aunque internamente el vehículo posee dicho motor.

- Laser medidor de distancias: el laser, situado en la parte frontal del vehículo, es capaz de detectar algún objeto que se interponga en su trayectoria y devolver la distancia a la que se encuentra dicho objeto.

- GPS: en el paquete ofrecido por RUC, también se encuentra un servicio que simula un GPS, aunque dicho GPS no posee los mapas por los cuales el vehículo va a circular. El GPS es de gran utilidad para la tarea de planificación de rutas, tarea que no entra dentro del entorno de este proyecto por lo que no se hará uso del GPS.

- Sensores de impacto: el vehículo de RUC posee cinco sensores de impacto, uno en cada lateral, otro en la parte delantera, uno más en la parte trasera y, por último, un quinto en el techo del vehículo. Estos sensores informarán de cualquier impacto si se ha producido en su zona, tanto de manera interna en el código, como de manera visual mostrándose en rojo en el interfaz suministrado por RUC y deteniendo el vehículo en ese mismo momento.

- Cámaras web: se suministran servicios para simular dos cámaras web situadas en los extremos delanteros del techo, tanto a la derecha como a la izquierda. Este es el elemento hardware simulado más importante para el ámbito de este proyecto, ya que se basa en la comprensión de la carretera mediante los datos obtenidos por las imágenes suministradas por las cámaras web.

Como ya se ha comentado en la introducción, el proyecto comienza en este punto, en el punto que ofrece la competición Robochamps y que ayuda a centrarse en lo verdaderamente importante para este proyecto, que es la creación de un sistema autónomo de control de un vehículo.

Por último, se debe comentar que la competición ha desaparecido por causas ajenas al proyecto, aunque en el momento de comenzar el mismo se disponía de soporte desde su página web. Actualmente, como se comenta, no existe dicho soporte ni la competición Robochamps.

2.- ESTADO DE LA CUESTIÓN

2.1.- ROBÓTICA

La robótica es el estudio de las maquinas que muestran cierto grado de autonomía y flexibilidad en las tareas que realizan (Chirikjian y otros, 2000).

Min Xie en su libro considera la robótica como una rama de la ingeniería que se encarga de los robots, los cuales los considera agentes físicos que son capaces de ejecutar acciones para la consecución de tareas (Xie, 2003).

En todo caso, según McKerrow (1991), se debe hacer una distinción entre ingeniería robótica y ciencia robótica. La ingeniería robótica se ocupa del diseño, construcción y aplicación de los robots. Mientras que los robots son construidos a través de la investigación que se realiza, la meta de la ciencia robótica no es desarrollar un robot, sino entender los procesos físicos y de información subyacentes a la percepción y acción.

Como se puede comprobar, la robótica es una disciplina muy amplia con un contenido significativo de áreas como física, matemáticas, ingeniería eléctrica, ingeniería mecánica y la informática. Como resultado de la investigación en las décadas de los setenta y ochenta, el volumen de conocimiento dentro de la robótica es amplísimo (McKerrow, 1991).

Más concretamente, la robótica se encarga de transformar el mundo real mediante un estado de entrada y obtener un estado de salida. Esta transformación se consigue mediante la utilización de elementos físicos que se encuentran en dicho mundo. De la definición de robótica como la conexión inteligente de percepción a acción se puede descomponer esta transformación física como una secuencia de procesos. A esta descomposición se le llama modelo de percepción de robots por que describe el proceso que se realiza con la información para percibir el estado actual del mundo y que el robot responda de manera adecuada. Estos procesos son medición, modelado, percepción, planificación y acción. Dichos procesos se encuentran implícitos en las arquitecturas o paradigmas que se verán más adelante. Aunque no se hable propiamente de estos procesos en cada una de estas arquitecturas, siempre se tienen en cuenta estas variables ya que de ellas depende el buen funcionamiento de un robot.

Los robots son usados en diversas aplicaciones, desde aplicaciones pedagógicas hasta las aplicaciones más industriales como pueden ser los brazos robóticos de las cadenas de montaje de automóviles. Cada aplicación posee su propio conjunto de problemas y su conjunto de requerimientos. Mientras que algunos robots se orientan más a mostrar novedades y evolución en esta disciplina, la introducción de los robots en las factorías es lo que realmente ha tenido un impacto considerable de la robótica en el mundo actual, si bien, este tipo de robots no son los que trataremos en este proyecto (McKerrow, 1991).

2.2.- HISTORIA DE LA ROBÓTICA

Por siglos, el ser humano ha construido máquinas que imitan las partes del cuerpo humano. Los antiguos egipcios unieron brazos mecánicos a las estatuas de sus dioses. Estos brazos fueron operados por sacerdotes, quienes clamaban que el movimiento de estos era inspiración de sus dioses. Los griegos construyeron estatuas que operaban con sistemas hidráulicos, los cuales se utilizaban para fascinar a los adoradores de los templos.

El inicio de la robótica actual puede fijarse en la industria textil del siglo XVIII, cuando Joseph Jacquard inventa en 1801 una máquina textil programable mediante tarjetas perforadas. La revolución industrial impulsó el desarrollo de estos agentes mecánicos, entre los cuales se destacaron el torno mecánico motorizado de Babbitt (1892) y el mecanismo programable para pintar con *spray* de Pollard y Roselund (1939). Además de esto durante los siglos XVII y XVIII en Europa fueron construidos muñecos mecánicos muy ingeniosos que tenían algunas características de robots. Jacques de Vaucansos construyó varios músicos de

tamaño humano a mediados del siglo XVIII. Esencialmente se trataba de robots mecánicos diseñados para un propósito específico: la diversión. En 1805, Henri Maillardert construyó una muñeca mecánica que era capaz de hacer dibujos. Una serie de levas se utilizaban como ' el programa ' para el dispositivo en el proceso de escribir y dibujar. Estas creaciones mecánicas de forma humana deben considerarse como inversiones aisladas que reflejan el genio de hombres que se anticiparon a su época.

La palabra robot se empleó por primera vez en 1920 en una obra de teatro llamada "R.U.R." o "Los Robots Universales de Rossum" escrita por el dramaturgo checo Karel Capek. La trama era sencilla: el hombre fabrica un robot luego el robot mata al hombre. Muchas películas han seguido mostrando a los robots como máquinas dañinas y amenazadoras. La palabra checa 'Robota' significa servidumbre o trabajador forzado, y cuando se tradujo a inglés se convirtió en el término robot.

Entre los escritores de ciencia ficción, Isaac Asimov contribuyó con varias narraciones relativas a robots, comenzó en 1939, a él se le atribuye popularizar el término “Robótica”. La imagen de robot que aparece en su obra es el de una máquina bien diseñada y con una seguridad garantizada que actúa de acuerdo con tres principios.

Estos principios fueron denominados por Asimov las Tres Leyes de la Robótica, y son:

1. Un robot no puede actuar contra un ser humano o, mediante la inacción, que un ser humano sufra daños.
2. Un robot debe de obedecer las órdenes dadas por los seres humanos, salvo que estén en conflictos con la primera ley.
3. Un robot debe proteger su propia existencia, a no ser que esté en conflicto con las dos primeras leyes.

Consecuentemente todos los robots de Asimov son fieles sirvientes del ser humano, de ésta forma su actitud contraviene a la de Capek.

Inicialmente, se definía un robot como un manipulador reprogramable y multifuncional diseñado para trasladar materiales, piezas, herramientas o aparatos a través de una serie de movimientos programados para llevar a cabo una variedad de tareas.

El desarrollo en la tecnología, donde se incluyen las poderosas computadoras electrónicas, los actuadores de control retroalimentados, transmisión de potencia a través de engranes, y la tecnología en sensores han contribuido a flexibilizar los mecanismos autómatas para desempeñar tareas dentro de la industria. Son varios los factores que intervienen para que se desarrollaran los primeros robots en la década de los 50's. La investigación en inteligencia artificial desarrolló maneras de emular el procesamiento de información humana con computadoras electrónicas e inventó una variedad de mecanismos para probar sus teorías.

Las primeras patentes aparecieron en 1946 con los muy primitivos robots para traslado de maquinaria de Devol. También en ese año aparecen las primeras computadoras: J. Presper Eckert y John Maulchy construyeron el ENAC en la Universidad de Pensilvania y la primera máquina digital de propósito general se desarrolla en el MIT. En 1954, Devol diseña el primer robot programable y acuña el término "autómata universal", que posteriormente recorta a Unimation. Así llamaría Engleberger a la primera compañía de robótica. La comercialización de robots comenzaría en 1959, con el primer modelo de la Planet Corporation que estaba controlado por interruptores de fin de carrera.

Figura 2: edificio principal del Massachusetts Institute of Technology (MIT)

En 1964 se abren laboratorios de investigación en inteligencia artificial en el Massachusetts Institute Technology (MIT), el Stanford Research Institute (SRI) y en la universidad de Edimburgo. Poco después los japoneses que anteriormente importaban su tecnología robótica, se sitúan como pioneros del mercado.

Otros desarrollos Importantes en la historia de la robótica fueron:

En 1960 se introdujo el primer robot "Unimate'', basada en la transferencia de artículos. Utilizan los principios de control numérico para el control de manipulador y era un robot de transmisión hidráulica.

En 1961 Un robot Unimate se instaló en la Ford Motors Company para atender una máquina de fundición de troquel.

En 1966 Trallfa, una firma noruega, construyó e instaló un robot de pintura por pulverización.

En 1971 El "Standford Arm'', un pequeño brazo de robot de accionamiento eléctrico, se desarrolló en la Standford University.

En 1973 Se desarrolló en SRI el primer lenguaje de programación de robots del tipo de computadora para la investigación con la denominación WAVE. Fue seguido por el lenguaje AL en 1974. Los dos lenguajes se desarrollaron posteriormente en el lenguaje VAL comercial para Unimation por Víctor Scheinman y Bruce Simano.

En 1978 Se introdujo el robot Programmable Universal Machine for Assambly (PUMA) para tareas de montaje por Unimation, basándose en diseños obtenidos en un estudio de la General Motors.

En 1980 Un sistema robótico de captación de recipientes fue objeto de demostración en la Universidad de Rhode Island. Con el empleo de visión de máquina el sistema era capaz de captar piezas en orientaciones aleatorias y posiciones fuera de un recipiente.

Actualmente, el concepto de robótica ha evolucionado hacia los sistemas móviles autónomos, que son aquellos que son capaces de desenvolverse por sí mismos en entornos desconocidos y parcialmente cambiantes sin necesidad de supervisión.

El primer robot móvil de la historia, pese a sus muy limitadas capacidades, fue Electro-Light-Sensitive Internal-External (ELSIE), construido en Inglaterra en 1953. ELSIE se limitaba a seguir una fuente de luz utilizando un sistema mecánico realimentado sin incorporar inteligencia adicional. En 1968, apareció SHACKEY del SRI, que estaba provisto de una diversidad de sensores así como una cámara de visión y sensores táctiles y podía desplazarse por el suelo. El proceso se llevaba en dos computadores conectados por radio, uno a bordo encargado de controlar los motores y otro remoto para el procesamiento de imágenes.

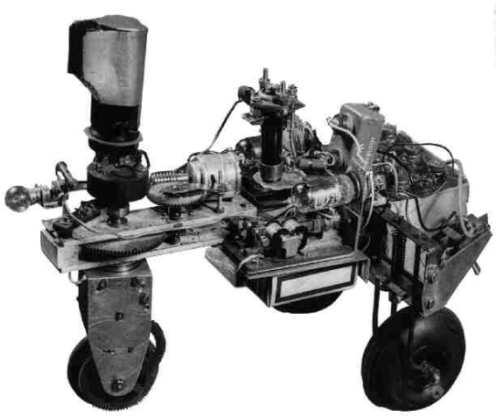


Figura 3: foto del primer robot de la historia, ELSIE

En los setenta, la NASA inicio un programa de cooperación con el Jet Propulsión Laboratory para desarrollar plataformas capaces de explorar terrenos hostiles. El primer fruto de esta alianza seria el MARS-ROVER, que estaba equipado con un brazo mecánico tipo STANFORD, un dispositivo telemétrico láser, cámaras estéreo y sensores de proximidad.

En los ochenta aparece el CART del SRI que trabaja con procesado de imagen estéreo, más una cámara adicional acoplada en su parte superior. También en la década de los ochenta, el CMU-ROVER de la Universidad Carnegie Mellon incorporaba por primera vez una rueda timón, lo que permite cualquier posición y orientación del plano.

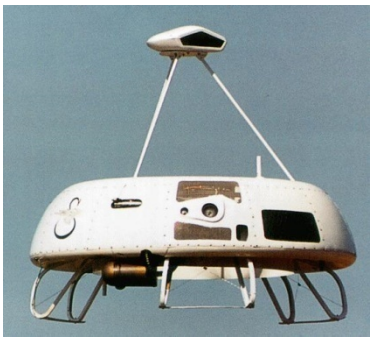


Figura 4: robot para uso militar CYPHER

En la actualidad, la robótica se debate entre modelos sumamente ambiciosos, como es el caso del IT, diseñado para expresar emociones, el COG, también conocido como el robot de cuatro sentidos, el famoso SOUJOURNER o el LUNAR ROVER, vehículo de turismo con control remotos, y otros mucho más específicos como el CYPHER, un helicóptero robot de uso militar, el guardia de tráfico japonés ANZEN TARO o los robots mascotas de Sony.

2.3.- ¿QUÉ ES UN ROBOT?

Hablando de robótica es imprescindible hablar de los verdaderos protagonistas, aparte del ser humano, los robots.

Muchas películas han seguido mostrando a los robots como máquinas dañinas y amenazadoras. Sin embargo, películas más recientes, como la saga de "La Guerra de las Galaxias" desde 1977, retratan a robots como "C3PO" y "R2D2" como ayudantes del hombre. "Número 5" de "Cortocircuito" y "C3PO" realmente tienen apariencia humana. Estos robots que se fabrican con *look* humano se llaman 'androides'.

La mayoría de los expertos en Robótica afirmaría que es complicado dar una definición universalmente aceptada. Las definiciones son tan dispares como se demuestra en la siguiente relación:

- Ingenio mecánico controlado electrónicamente, capaz de moverse y ejecutar de forma automática acciones diversas, siguiendo un programa establecido.

- Máquina que en apariencia o comportamiento imita a las personas o a sus acciones como, por ejemplo, en el movimiento de sus extremidades.

- Un robot es una máquina que hace algo automáticamente en respuesta a su entorno.

- Un robot es un puñado de motores controlados por un programa de ordenador.

- Un robot es un ordenador con músculos.

Es cierto, como acabamos de observar, que los robots son difíciles de definir. Sin embargo, no es necesariamente un problema el que no esté todo el mundo de acuerdo sobre su definición. Quizás, Joseph Engelberg (padre de la robótica industrial) lo resumió inmejorablemente cuando dijo: "Puede que no sea capaz de definirlo, pero sé cuándo veo uno".

La imagen del robot como una máquina a semejanza del ser humano, subyace en el hombre desde hace muchos siglos, existiendo diversas realizaciones con este fin. Aunque se persiga este hecho, hay distintos tipos de robots para utilizarlos en distintas situaciones.

Figura 5: robot RX, uno de los más modernos que existen

Atendiendo a distintas características de cada uno de los robots se pueden realizar infinidad de clasificaciones, existen robots androides que es el nombre con el que se identifica a los robots que se parece al ser humano e intentan actuar como él, robots zoomórficos que, como su nombre indica, simulan las formas animales ya sea mediante las patas o distribución del cuerpo.

Uno de los tipos de robots más comunes en la sociedad actual son los brazos robóticos industriales. Estos brazos robóticos, ya sean autónomos o movidos por un ser humano son muy útiles a la hora de realizar tareas y sobre todo tareas repetitivas donde el ser humano puede perder eficiencia debido al cansancio. Actualmente, la robótica ha tenido su mayor funcionalidad plasmada en estos robots pero se continua investigando para que no todo sean tareas repetitivas, sino para conseguir lo más parecido a un ser humano

En el ámbito del proyecto, el tipo de robot que se utilizará será un robot móvil y, más concretamente, autónomo móvil. Dichos robots se trataran de forma general.

2.4.- ARQUITECTURAS ROBÓTICAS

Todos los robots se construyen sobre arquitecturas de control, es decir, para construir un buen robot es necesario seguir algunas de las arquitecturas conocidas. De manera más técnica, se podría decir que una arquitectura de control es un sistema software que establece las acciones o movimientos que debe realizar el robot a partir de la adquisición y tratamiento de la información sensorial y del objetivo u objetivos que le hayan sido indicados (Lope, 2006).

Para (Arkin, 1999) la "arquitectura robótica es la disciplina dedicada al diseño de robots específicos individuales mediante bloques comunes de desarrollo software".

Según muestra L. Enrique Sucar (2008) en su introducción a la robótica, una arquitectura robótica es la "organización de la generación de acciones a partir de las percepciones del robot".

Algunos autores, en vez de hablar de arquitecturas, se refieren a este campo como "paradigma" (Murphy, 2000). Según Robin Murphy, un paradigma es "una filosofía o conjunto de hipótesis y/o técnicas que caracterizan una aproximación a un tipo determinado de problemas". Y es que las arquitecturas que describiremos a continuación están basadas en los paradigmas.

Por lo tanto, mediante el conocimiento de las arquitecturas robóticas se consigue una de las claves a la hora de solucionar problemas relacionados con la inteligencia artificial (IA) y los robots.

Las arquitecturas de control poseen una serie de características comunes que deben cumplir para lograr ser cien por cien eficientes y que, según Javier de Lope (2006) se pueden enumerar de la siguiente manera.

- Capacidad de abordar múltiples objetivos de forma simultánea.
- Capacidad de integración de la información de múltiples sensores de diferente procedencia.
- Robustez ante fallos de elementos del sistema.
- Adaptación ante nuevos entornos.
- Capacidad de extensión y modificación a lo largo de su vida.
- Capacidad para considerar posibilidades y valorar consecuencias.
- Interacción con el entorno para percibir cambios en él y responder adecuadamente.

Estas características hacen que el robot consiga un dinamismo tal, que gracias a él será capaz de aprender y desenvolverse de manera idónea en un entorno a priori desconocido.

Otro punto a comentar es la elección de unas arquitectura u otra la cual y, como bien indica Ramón Ignacio Barber Castaño en su tesis doctoral (2000), "depende enteramente de la comparación de ambas filosofías bajo el marco de una tarea determinada, del entorno y de las capacidades computacionales y sensoriales del robot". Por lo que no existe una normativa o regla en la que se marqué qué arquitecturas hay que utilizar para determinadas acciones o para determinados tipos de robot, depende de muchos factores que la persona desarrolladora debe valorar y en base a ellos decidir el tipo de arquitectura, aunque sí que se presentan características comunes que nos pueden ayudar a decidirnos por una arquitectura u otra.

Para Robin Murphy, los paradigmas se pueden dividir en dos clasificaciones según su comportamiento.

Por la relación de tres elementos comúnmente conocidos en robótica: detección-planificación-actuación. Como su propio nombre indica, las funciones de un robot se dividen en tres partes. En la parte detección, los sensores recogen los datos del entorno. El apartado de planificación se encarga de decidir qué acción realizar con los datos recogidos de los sensores y los datos almacenados del mundo que rodea al robot. Y, por último, en el apartado de actuación, el robot ejecuta la acción decidida.

Por la manera en que la información recibida de los sensores es procesada y distribuida a través del sistema: o cuanto afecta al humano, animal o máquina lo que recibe por los sensores. Es decir, la reacción que se ofrece cuando se recibe información del exterior por medio de los sensores.

Teniendo en cuenta esta división creada por Robin Murphy, existen tres paradigmas sobre los que se basan las arquitecturas actuales: paradigma jerárquico (deliberativo, planificado), paradigma reactivo y, aunque no es un paradigma nuevo propiamente dicho, paradigma híbrido que, como su propio nombre indica, es una mezcla de los dos primeros paradigmas.

Para que estos paradigmas sean efectivos en el mundo real, las arquitecturas creadas se basaran en estos. La arquitectura simplemente proporciona las herramientas y el desarrollo necesario para llevar a una realidad el paradigma en el robot.

2.4.1.- ARQUITECTURAS JERÁRQUICAS

Estas arquitecturas, también llamadas arquitecturas jerárquicas (Murphy, 2000) o arquitecturas planificadas (Barber Castaño, 2000), siguen un patrón claro detección-planificación-actuación. Es decir, es una arquitectura en la que primero detecta el entorno, posteriormente decide (delibera) la acción a realizar y por último realiza la acción decidida.

La arquitectura deliberativa se basa en enfoques tradicionales de la inteligencia artificial (IA) simbólica fundamentándose en la hipótesis de que el comportamiento inteligente puede ser logrado mediante un sistema de símbolos que permite la composición y tratamiento de los mismos que representan entidades del mundo real. Su filosofía podría ser resumida como “pensar mucho y luego actuar” (Lope, 2006).

Para Barber Castaño (2000), estas arquitecturas se basan en dos fundamentos básicos:

- La descomposición funcional de la arquitectura: aporta la descomposición de la tarea en un refinamiento progresivo hasta que se reduzca al máximo el nivel de abstracción del problema, para tener más próxima la solución del mismo.

- La generación de un modelo de mundo: se general este modelo de mundo en base a símbolos que son almacenados. A la hora de deliberar la acción a realizar, se consulta con los símbolos del mundo que se ha creado. Se podría presentar el problema de que si el mundo es cambiante mientras generamos estos símbolos, jamás vamos a tener un modelo de mundo fiel a la realidad, pero lo cierto es que este tipo de arquitecturas son utilizadas en mundos estáticos por lo que no presenta gran problema.

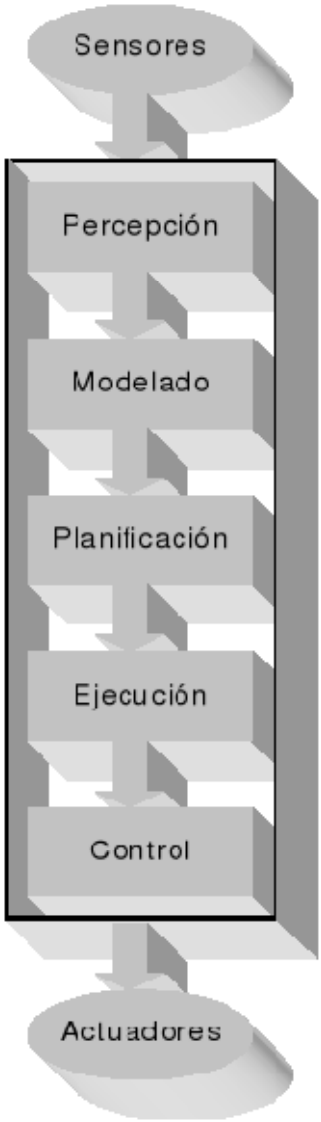


Figura 6: niveles de una arquitectura de descomposición funcional (Lope, 2006).

Se caracteriza por tener una representación interna central (RIC) con una descripción simbólica del ambiente que le rodea, las acciones del agente y sus objetivos. Además posee otros módulos como pueden ser planificación, aprendizaje, etc. que se comunican entre sí a través de RIC. Con lo que el RIC es el punto de paso para cualquier información recibida de los sensores. El razonamiento de la acción a ejecutar se realiza con un "*planning*" definido previamente el cual indicará al robot qué debe hacer para lograr alcanzar el objetivo. La ejecución de los planes, en su gran mayoría, es de lazo abierto, es decir, la salida no alimenta ninguna entrada.

Las ventajas de estas arquitecturas son que tenemos la habilidad para alcanzar objetivos de alto nivel de complejidad evitando errores y permite determinar planes de acción óptimos.

Las desventajas mostradas por este tipo de arquitecturas son lentitud en el procesamiento de las tareas a realizar, ya que suelen estar basadas en técnicas de búsqueda muy pesadas y en ambientes no deterministas y cambiantes es necesario realizar añadidos para adaptarse a ese tipo de ambiente. De hecho, este tipo de arquitectura no suele ser utilizada cuando nos encontramos en un entorno cambiante.

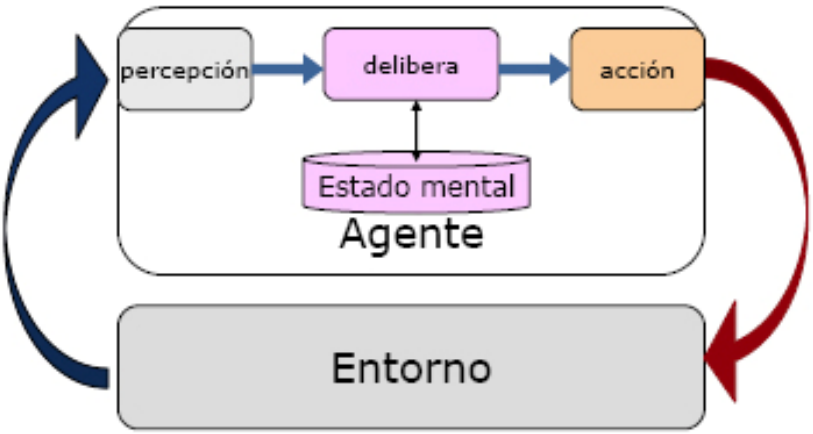


Figura 7: estructuración de la arquitectura deliberativa (Pavón Mestras, 2006).

Un ejemplo de arquitectura deliberativa, jerárquica o planificada es la arquitectura Nasa Standard Reference Model (NASREM).

Esta arquitectura creada por James S. Albus representó la culminación de 15 años de trabajo por parte del National Institute of Standards and Techonolgy (NIST) en la investigación de los sistemas en tiempo real para robots y máquinas inteligentes. En 1987 fue la primera vez que se utilizó esta arquitectura con un sistema en tiempo real.

En esta arquitectura, el sistema de control es representado como tres niveles jerárquicos de módulos de computación ayudados por un sistema de comunicación y una memoria global.

El primer modulo es la descomposición de tareas que planifica y ejecuta la descomposición de las metas iniciales en metas más pequeñas y manejables para el sistema. Este modulo implica simultáneamente una descomposición temporal y una descomposición espacial.

El segundo módulo de la jerarquía es el modelado del mundo o entorno el cual se encarga de modelar y evaluar el entorno que rodea al sistema. El modelo de mundo es el mejor sistema de estimación y evaluación de la historia, estado actual y posibles estados futuros de nuestro entorno. Por supuesto, el conocimiento y mantenimiento de la información del entorno se realiza gracias a los sensores que obtienen la información y la introducen en el sistema.

Y el último nivel de la jerarquía es el sistema de sensores. Se encarga de reconocer patrones, detectar eventos y filtrar e integrar la información del espacio y tiempo. Este modulo se encarga de comparar las predicciones hechas del sistema sobre el mundo real con las informaciones que se obtienen de los sensores y computa funciones de similitud y de diferencia. Por supuesto, este modulo se encarga de recoger los objetos, eventos y patrones que detecta en el entorno y enviarlos al modulo de modelización del entorno.

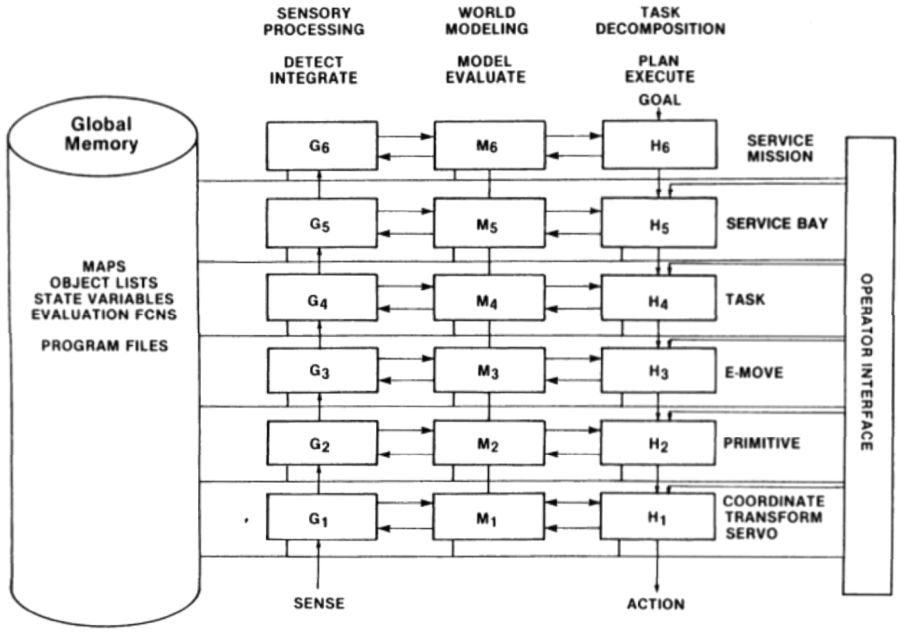


Figura 8: arquitectura de control NASREM.

2.4.2.- ARQUITECTURAS REACTIVAS

La arquitectura reactiva, o paradigma reactivo, nació en los años ochenta para, posteriormente, ser el paradigma que siente las bases de otro paradigma, muy utilizado comúnmente, que es el hibrido. Además, los sistemas robóticos que tienen un dominio de tareas limitado todavía son construidos siguiendo este paradigma.

Para (Friedenberg, y otros, 2006), el paradigma reactivo “nace de la idea de que los comportamientos complejos surgen de los comportamientos simples y que operan de manera concurrente”.

Por otro lado este paradigma nace de la insatisfacción que el paradigma jerárquico (deliberativo) junto con una mezcla de nuevas ideas. Esta insatisfacción fue mostrada, principalmente, por Rodney Brooks el cual caracterizaba las arquitecturas jerárquicas como una “descomposición horizontal” (Murphy, 2000).

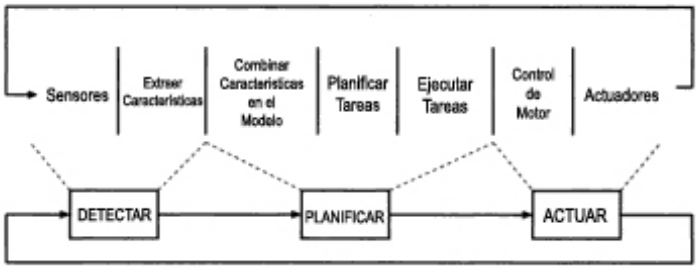


Figura 9: idea de Rodney Brooks sobre el paradigma jerárquico (Murphy, 2000).

En lugar de esto, un estudio etimológico de lo que significaba inteligencia sugirió que dicha inteligencia se estructura de forma vertical. Bajo esta descomposición vertical, un agente comienza con primitivos comportamientos de supervivencia y evoluciona a nuevos tipos de comportamiento, desde los más simples hasta los más avanzados. Cada uno de estos comportamientos tiene acceso independiente a los sensores y actuadores, de manera que, si algo sucede con los comportamientos avanzados, los básicos seguirán funcionando de manera correcta (Murphy, 2000).

De manera concreta, a la hora de realizar un sistema robótico operativo, los comportamientos se realizan en capas que son independientes y que, como se ha comentado, tienen acceso a los sensores y actuadores.

Al principio se dudaba de la eficacia de este paradigma. Los usuarios de robótica no estaban conformes con la imprecisa manera en la que los comportamientos discretos se combinaban con los comportamientos más potentes. Finalmente, los rápidos tiempos de ejecución asociado a unos comportamientos reflexivos terminaron por convencer a la sociedad robótica y a posteriori surgiría el paradigma híbrido, del que se hablará más adelante.

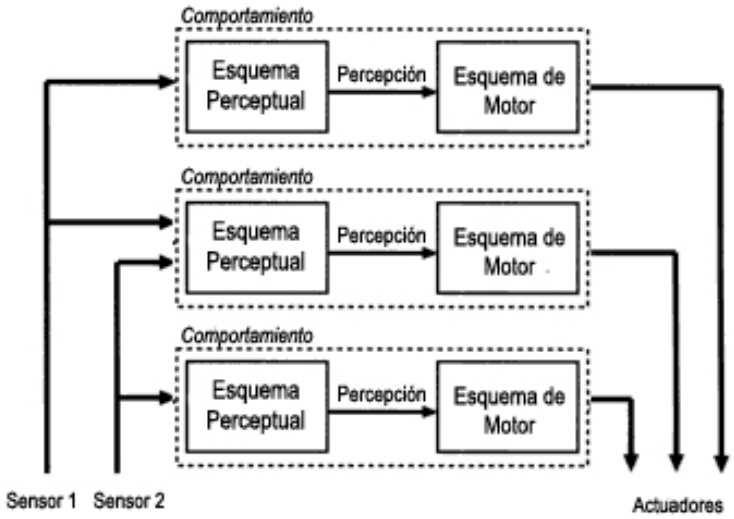


Figura 10: descomposición vertical de tareas (comportamientos) asociada al paradigma reactivo (Murphy, 2000).

Asemejando el paradigma reactivo con la estructura jerárquica detección-planificación-actuación, se podría decir que el reactivo omite la parte de planificación, solo utiliza detección-actuación. Las arquitecturas creadas a partir del paradigma reactivo son las encargadas de especificar cómo se coordinan y controlan los comportamientos.

Las características básicas se podrían resumir en (Arkin, 1999):

- Énfasis en la importancia del binomio detección-actuación.
- Evitar toda representación simbólica del conocimiento.
- Descomposición en unidades o comportamientos.

Llegados a este punto y, una vez que se ha introducido el paradigma reactivo, es obligatorio tratar sobre la arquitectura que más se ajusta a este paradigma, la arquitectura de subsunción de Rodney Brooks.

Rodney Brooks desarrolló esta arquitectura a mediados de los ochenta en el MIT argumentando que el paradigma jerárquico y sus arquitecturas evolucionaban en detrimento de la construcción de robots de trabajo reales. Decía que la construcción de modelos de mundo y razonamientos usando representaciones simbólicas del conocimiento era un impedimento para el tiempo de respuesta de un robot y llevaba a los investigadores de robótica por el camino equivocado (Arkin, 1999).

Según (Murphy, 2000), la arquitectura de subsunción de Rodney Brooks es la arquitectura basada en el paradigma reactivo más influyente en el mundo de la robótica. En parte, esta influencia fue debida a la publicidad creada de los sistemas generados con esta arquitectura. Los robots creados a partir de esta arquitectura lucen como insectos del tamaño de una caja de zapatos, con seis patas y antenas. Además, estos robots fueron los primeros capaces de caminar, evitar colisiones y trepar sobre los obstáculos sin las pausas típicas “mover-pensar-mover-pensar” de los antiguos robots.

El termino comportamiento en la arquitectura de subsunción de Brooks tiene un significado menos preciso que en otras arquitecturas. Un comportamiento es una red de módulos de actuación y detección que realizan una tarea. Estos módulos son máquinas finitas de estados aumentadas las cuales tienen registros, tiempos y otras mejoras que les permiten interactuar con otros módulos (Murphy, 2000). Se puede decir que la máquina de estados finita aumentada es el equivalente a la interfaz entre los esquemas y la estrategia de coordinación de control en un esquema de comportamientos.

Existen cuatro aspectos interesantes cuando hablamos de esta arquitectura:

- Los módulos son agrupados en capas. Las capas reflejan una jerarquía de inteligencia. Las capas de bajo nivel encapsulan funciones de supervivencia básicas mientras que los niveles más altos crean acciones para la consecución de metas. Cada capa puede ser vista como un comportamiento abstracto para una tarea en particular.

- Los módulos que están en capas altas pueden ser sobrescribir o suprimir la salida por comportamientos que están en la capa inmediatamente anterior. Las capas de comportamiento actúan de manera independiente y concurrentemente por lo que es necesario un mecanismo para manejar los potenciales conflictos. La solución en subsunción es que el ganador es siempre la capa más alta.

- Se evita el uso de estados internos. Se entiende estado interno como cualquier tipo de representación persistente local que muestre el estado del mundo. Como el robot es un agente situado, la gran mayoría de esta información es proporcionada por el mismo entorno en tiempo real.

- Una tarea es completada debido a la activación de la capa que corresponde, la cual activa capas de menor nivel y así continuamente. Sin embargo, los sistemas de subsunción no son fácilmente divisibles en tareas, es decir, no se puede ordenar que se realice otra tarea sin ser reprogramada.

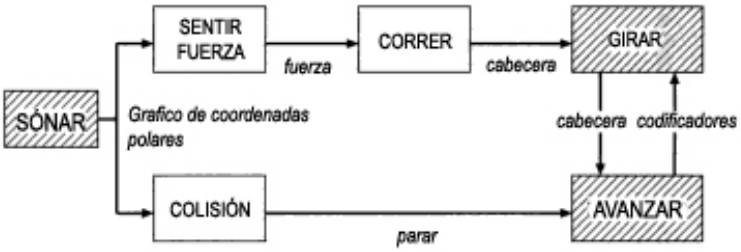


Figura 11: nivel 0 de la arquitectura de subsunción (Murphy, 2000).

2.4.3.- ARQUITECTURAS HÍBRIDAS

Hacia finales de los ochenta, la tendencia en inteligencia artificial de robots era diseñar y programar basándose en el paradigma reactivo. Pero el coste de aplicar el paradigma reactivo era que debías eliminar la planificación o cualquier otra función que involucrara memorizar o razonar sobre el estado global del robot en relación a su entorno. Esto significa que un robot no podría planear una ruta optima, crear mapas, monitorizar su propia configuración o seleccionar el comportamiento más adecuado para llevar a cabo la tarea indicada.

Por lo que en los inicios de los años noventa, la comunidad de IA se encontraba en un punto en el que la deliberación y la planificación de rutas habían vuelto al panorama robótico pero, al mismo tiempo, no se debía perder en gran logro que supuso el paradigma reactivo (Murphy, 2000). Además, la búsqueda de los paradigmas robóticos es emular los comportamientos humanos que encontramos a diario, esto es, era imprescindible contar con un sistema reactivo (Arkin, 1999).

Debido, principalmente a estas razones expuestas arriba, se hizo necesaria la creación de un paradigma híbrido.

El actual pensamiento de la comunidad robótica es que el paradigma híbrido es la mejor solución general por varias razones. Utiliza técnicas de procesamiento asíncronas permitiendo a las funciones deliberativas ejecutarse independientemente de los comportamientos reactivos (Murphy, 2000).

De hecho, era más que inevitable que, teniendo en cuenta que ambos paradigmas, tanto el reactivo como el deliberativo, contaban con ventajas que cubrían los inconvenientes del otro paradigma, surgiera un paradigma híbrido. La primera fue Mataric, alumna de Brooks, la cual incluyó una capa de planificación en la parte superior del sistema de capas múltiples del paradigma reactivo (Bekey, 2005).

La organización de un sistema híbrido se podría describir como planificar, y después detectar-actuar. El apartado de planificación incluye toda la deliberación y el modelado de un mundo global, no solo tareas y planificación de rutas, es decir, el robot, primero, debe pensar (planificar) como va a conseguir la misión o tarea y después instancia una serie de comportamientos para ejecutar el plan o una parte de él.

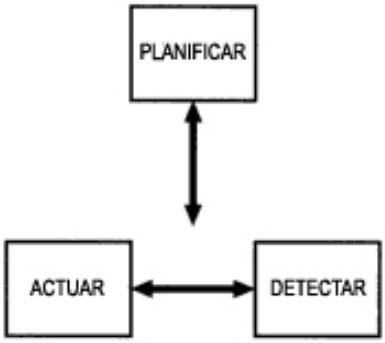


Figura 12: organización del paradigma híbrido

La organización de las primitivas detección, actuación y planificación están divididas conceptualmente en una parte reactiva y otra parte deliberativa. Aunque muchas arquitecturas tienen discretas capas de funcionalidades dentro de la parte reactiva y deliberativa, cada arquitectura tiene la parte reactiva y la parte deliberativa.

Según (Murphy, 2000) existen tres categorías de arquitecturas híbridas:

- Estilo gestionado: enfocado en la subdivisión de la parte deliberativa en capas basadas en el enfoque de control de cada función deliberativa.
- Jerarquías de estado: usan el conocimiento del estado del robot para distinguir entre comportamientos reactivos y actividades deliberativas. Los comportamientos reactivos se ven sin estado y funciones solo de presente. Las funciones deliberativas pueden ser divididas entre las que necesitas un conocimiento de un estado pasado, anterior y las funciones que requieren conocimiento de un estado futuro.
- Estilos orientados al modelo: Se caracterizan por que los comportamientos tienen acceso a partes del modelo de mundo.

2.4.4.- ARQUITECTURAS APLICADAS AL PROYECTO

En relación al proyecto de fin de carrera, se han ido utilizando dos arquitecturas principales. Primero, para tomar contacto con los elementos que presentaba el proyecto se siguió una arquitectura reactiva debido a que, probablemente, sea la arquitectura más sencilla y rápida de implementar y que ofrece unos resultados nada despreciables aunque no idóneos para la solución buscada.

Una vez analizadas todas las variables ofrecidas, obtenida la experiencia necesaria y observando que los resultados no eran todo lo deseables que se desearía, de manera automática y natural, se hizo necesario cambiar de arquitectura. EL cambio se produjo hacia una arquitectura híbrida, aprovechando algunos de los elementos realizados en la anterior fase reactiva.

Como se analizará en otros apartados de este documento, las razones para cambiar esta arquitectura están basadas en números y porcentajes y en la necesidad de tener un apartado en el cual se tome decisiones mediante una deliberación más trabajada y exacta.

Gracias a este proyecto se ha hecho fehaciente que los avances en este apartado de la robótica fueron necesarios y acertados y que, efectivamente, una arquitectura no está por encima de la otra, sino que cada una es apropiada para distinto tipo de problemas y que, incluso, pueden coexistir en un mismo proyecto. Teniendo esto en cuenta, es necesario decir que ninguna arquitectura representa la solución universal a todos los problemas, sino que dependiendo de la situación se debe emplear una arquitectura u otra o ambas.

2.5.- LA ROBÓTICA COGNITIVA

La palabra *cognición* corresponde a una etimología latina de los términos *conocimiento* y *conocer*. Según el diccionario de la Real Academia Española, *conocer* es *"averiguar por el ejercicio de las facultades intelectuales, la naturaleza, cualidades y relaciones de las cosas"*[1]. Obviamente debemos aplicar esta definición al área de los robots y lograr que dichas máquinas logren actuar de manera simulada como humanos en el proceso del conocimiento.

Según Neisser (1976), cualquier cosa que conozcamos acerca de la realidad, tiene que ser mediada, no sólo por los órganos de los sentidos, sino por un complejo de sistemas que interpretan y reinterpretan la información sensorial. Con lo cual, el termino *cognición* es definido como los procesos mediante los cuales el input sensorial es transformado, reducido, elaborado, almacenado, recobrado o utilizado.

Los procesos cognitivos básicos son sensación, percepción, atención o concentración y memoria. Los procesos cognitivos avanzados son pensamiento, lenguaje e inteligencia.

La sensación es la reacción a la obtención de estímulos provocados por agentes externos. Aplicado a la robótica, los sensores, laser y otros componentes que analizan el entorno serán los que proporcionen esta sensación a los robots.

La percepción consiste en organizar e interpretar esa información que se ha captado con anterioridad. Nuevamente, si aplicamos este proceso cognitivo a la robótica vemos que el robot debe ser capaz de entender la información que se ha captado en los elementos que nos proporcionan datos del entorno.

La atención o concentración es la capacidad que permite centrarse en la información que en ese momento se considere primaria durante un espacio de tiempo. Un robot deberá saber qué es más importante en cada momento para lograr reaccionar de manera satisfactoria a los estímulos recibidos del exterior, por ejemplo, un posible impacto con una farola a 10 metros o un semáforo en rojo a 2 metros. Mediante este proceso el robot, aunque capte el impacto a 10 metros debe saber que la prioridad está en no saltarse el semáforo en rojo y se debe detener.

La memoria es el proceso por el cual se es capaz de recuperar información almacenada en un pasado y reutilizarla en el presente, de manera que no es necesario aprender dos veces lo mismo. Aplicado este proceso a un robot, la máquina debe ser capaz de memorizar ciertos elementos o situaciones que han sucedido para poder elaborar un proceso de selección en el caso de que se encuentre en una situación parecida o incluso exacta.

Los procesos cognitivos superiores son los más difíciles de recrear ya que requieren de un comportamiento innato a los humanos, un comportamiento difícilmente expresable en número o palabras, que no sigue determinados patrones y que es único en cada humano. Simular estos procesos en robots es el gran desafío al que se enfrenta la comunidad robótica mundial.

Por todo ello, la investigación en robótica cognitiva se centra en proporcionar a los robots funciones cognitivas superiores que les permita razonar, actuar y percibir de manera robusta en entornos desconocidos y cambiantes. Se dice que esta rama de la robótica intenta lograr que los robots adquieran una consciencia artificial, algo que los humanos poseemos de manera innata y que se corresponde con un conjunto de procesos cognitivos. La consciencia se refiere al conocimiento de uno mismo, conocimiento de su situación en el espacio y su estado. Pero la aplicación de estos términos a un robot resulta casi imposible, por ello se entiende la consciencia como el estado cognitivo no abstracto que permite la interactuación, interpretación y asociación con los estímulos externos denominados realidad.

[1] Definición de la palabra conocer: RAE:
http://buscon.rae.es/draeI/SrvltConsulta?TIPO_BUS=3&LEMA=Conocer

Una vez que se ha introducido los procesos cognitivos de los humanos, es necesario explicar qué influencia han tenido estos procesos en el proyecto desarrollado.

Como se ha comentado en otros apartados, continuamente se está buscando desarrollar el robot que sea más fiel al comportamiento humano por varias razones. Algunas de las razones por la que se busca esta similitud es para adaptar dichos robots a la sociedad e integrarlos en ella en un futuro. Otras razones son la búsqueda de las cualidades que los humanos tenemos para la realización de ciertas tareas.

Por otro lado, hay procesos cognitivos que un robot es capaz de realizar con mayor precisión, como la gestión de ciertos tipo de memoria, que si bien un humano puede olvidar o no recordar momentáneamente ciertas cosas, el robot siempre tendrá acceso a la memoria que tenga almacenada, siempre y cuando se haya almacenado correctamente. En ese caso, el cien por cien de las veces el robot será capaz de obtener y mostrar dicha información y de manera exacta a como estaba cuando se almacenó, cosa que una persona no puede garantizar.

La misma situación ocurre con el proceso de atención. Simplemente, el robot realiza el trabajo para el que está desarrollado y todo lo que esté fuera del entorno de su trabajo será omitido. Por esta cualidad, se intentan incorporar robots industriales que son más eficientes en la realización de tareas repetitivas como las que se pueden desarrollar en cadenas de montaje, fábricas, etc.

Para el proyecto, el proceso cognitivo que se trata es el de la percepción de una curva e interpretación del giro necesario, intentando anticipar el movimiento a la situación con la que el robot se va a encontrar. Para ello el humano realiza una adquisición de datos de manera extremadamente rápida, interpreta que lo próximo con lo que se va a encontrar es una curva y estima qué grado de giro necesitará. Naturalmente, este grado de giro puede ser erróneo, con lo que el humano rectifica y aumenta el grado de giro o lo disminuye según lo necesite.

Este proceso, tan trivial para los humanos, es algo más complicado de realizar en un robot, puesto que el robot no sabe qué es una curva, o qué es un grado de giro, todo debe conocerlo para poder reaccionar correctamente. El robot, tomando el anterior caso como ejemplo, recogerá los datos obtenidos de las líneas que le indican qué situación tiene la carretera. Analizando esas líneas obtendrá el patrón que actualmente está siguiendo la carretera (recta, curva, etc.). Desgraciadamente el robot no puede anticipar una curva con cincuenta metros de distancia, lo que es una desventaja, pero sin embargo, es capaz de gestionar y comprobar datos extremadamente rápido, con lo que aunque la curva que se debe tomar esté a diez metros, el ordenador será capaz de realizar todos los cálculos necesarios para interpretar que tiene una curva y que debe girar y, además, calcular el grado de giro que debe aplicar. Al igual que un humano, este grado de giro se debe volver a calcular continuamente puesto que es necesario realizar rectificaciones en el giro en la mayor parte de los casos.

Este proceso es un ejemplo de la inspiración en los procesos cognitivos humanos que se ha tomado en el proyecto. El objetivo principal, como se ha comentado, es el de incorporar procesos que los humanos realizamos a diario y de manera natural, instintiva, en un robot que no conoce nada de dichos procesos y que solo captura datos de manera continua. Para ello es imprescindible una serie de hardware asociado que sustituya los sentidos que el ser humano posee.

En el ámbito del proyecto, es absolutamente necesario e imprescindible el sentido de la vista. Este sentido es simulado con dos cámaras web situadas en el vehículo que hacen de “ojos” para el robot. Mediante estos “ojos” es capaz de capturar la información para posteriormente gestionarla internamente y obtener los datos necesarios. Por supuesto, todo este proceso debe ser desarrollado ya que, aunque las cámaras web capturen datos, si no se desarrolla un sistema de obtención e interpretación de los datos de dichas imágenes, el robot no realizaría ninguna acción.

2.6.- ROBOTS AUTÓNOMOS MÓVILES

Dentro de los robots comentados anteriormente, en este proyecto se hará especial hincapié en los robots móviles autónomos ya que será el tipo de robot que se utilizará en el proyecto, virtualmente, pero de este tipo.

Como su nombre indica, un robot autónomo móvil o robot móvil autónomo es un robot con capacidad de movimiento, ya sea por ruedas, piernas u otro sistema de locomoción, que posee un cierto grado de autonomía.

Más concretamente, la propiedad de autonomía que, se puede decir, es el concepto introducido en este tipo de robots, se entiende como la habilidad para tomar decisiones inteligentes en una situación cambiante (Meystel, 1991). Cuando se habla de autonomía, se habla de que cada robot posee cierto grado de la misma, evidentemente, no existe ningún robot cien por cien autónomo ya que, como mínimo, debe ser puesto en marcha por un humano. Al hablar de autonomía hablamos de que el robot hace ciertas tareas sin la ayuda de nadie, solo con su propia toma de decisiones y sus movimientos.

Estos robots los encontramos en varias áreas como sistemas de manufactura, automatización de sistemas de oficina, mantenimientos de sistemas y sistemas de seguridad, y por supuesto, se siguen extendiendo.

Para entenderlo mejor, se va a estudiar cada uno de los principales elementos que componen un robot, tanto elementos físicos como la locomoción, como elementos internos del hardware.

2.6.1.- LOCOMOCIÓN

Un robot autónomo móvil necesita un medio de locomoción que lo haga realmente móvil. Para ello se dispone de múltiples soluciones. Muchas de ellas están basadas en recursos que intentan asemejar lo que existe en la vida real, es decir, sistemas de movimiento mediante piernas, patas, etc. Encontramos una excepción, la cual no es un recurso biológico, sino que es un recurso inventado por el hombre, la rueda.

Para (Siegwart, y otros, 2004), los mecanismos de locomoción deben solucionar al mismo tiempo problemas de estabilidad, características de contacto y adaptarse al entorno que le rodea.

- Estabilidad
  - Numero y geometría de los puntos de contacto
  - Centro de gravedad
  - Estabilidad estática/dinámica
  - Inclinación del terreno

- Características de contacto
  - Forma y tamaño del punto de contacto
  - Angulo de contacto
  - Fricción

- Tipo de entorno
  - Estructura
  - Medio

Figura 13: distintos tipos de locomoción para robots móviles autónomos

2.6.2.- CINEMÁTICA

La cinemática es el estudio más básico de cómo se comportan los sistemas mecánicos. En robótica móvil necesitamos entender el comportamiento de un mecanismo robótico para realizar robots que se ajusten al máximo a las tareas que en un futuro van a realizar y para entender cómo crear el software de control para dicho robot.

Cuando analizamos la cinemática de un robot móvil principalmente estamos hablando de robots móviles equipados con ruedas. El proceso de comprensión de la cinemática de un robot comienza con el proceso de descripción de la contribución de cada rueda al movimiento. Cada rueda juega un rol en el movimiento de un robot móvil y a su vez tiene sus restricciones a la hora de mover la máquina.

Los robots autónomos móviles que se simularan en este proyecto se mueven sin dirección alguna, simplemente cambian la potencia ejercida en cada rueda, de tal manera que si queremos girar a la derecha debemos aplicar más potencia sobre la rueda izquierda y viceversa s queremos girar a la izquierda.

2.6.3.- PERCEPCIÓN

Una de las tareas más importantes de un sistema autónomo de cualquier tipo es la adquisición del conocimiento sobre su entorno. Esta tarea se realiza mediante la toma de medidas usando varios sensores y extrayendo la información que le interesa al robot de las medidas tomadas.

Hay una gran variedad de sensores que se usan en robótica móvil. Muchos sensores son usados como un simple medidor de variables del entorno. Algunos otros sensores, más sofisticados, pueden ser utilizados para obtener información sobre el entorno del robot o para hallar la posición global del robot.

En este ámbito, los sensores se pueden clasificar de dos maneras: activos y pasivos o propioceptivos y exteroceptivos (Siegwart, y otros, 2004)

Los sensores propiocetivos miden valores internos del sistema robótico. Un ejemplo de estas medidas puede ser la velocidad del motor o el voltaje de la batería.

Los sensores exteroceptivos adquieren información del entorno del robot, como puede ser la intensidad de la luz, la amplitud del sonido entre otros.

Los sensores pasivos se encargan de medir la energía ambiental que entra en los sensores, esto es, mediciones de variables ambientales que se encuentran en el entorno del robot como pueden ser temperatura, sonidos (mediante un micrófono), etc.…

Lo sensores activos emiten energía al entorno para provocar una reacción del entorno y tomar la medida del mismo. Como este tipo de sensores pueden gestionar más interacciones controladas con el entorno, generalmente suelen lograr un mayor rendimiento en la toma de datos.

2.6.4.- LOCALIZACIÓN

Dentro de lo que conforma la navegación de un robot hay cuatro elementos. Lograr una buena navegación requiere lograr un buen desarrollo de estos cuatro elementos. Estos elementos son percepción, localización, cognición y control de movimiento.

De estos cuatro elementos la localización es el que ha recibido el mayor desarrollo. Como se puede adivinar, el sistema de localización de un robot se encarga de detectar la posición global y relativa que el robot tiene en ese momento. Por supuesto, para este fin el robot debe ayudarse de sensores y demás soporte hardware. En el caso de poseer un GPS con el mapa del entorno, la localización global se hace extremadamente sencilla, desgraciadamente, en la gran mayoría de los casos nunca sabremos el mapa del entorno con que el robot deberá aprender el entorno y almacenarlo con posiciones relativas para que cuando vuelva a esa misma posición el robot sepa que ya ha estado en ese punto antes.

2.6.5.- PLANIFICACIÓN Y NAVEGACIÓN

Hasta ahora, hemos tratado apartados que estaban enfocados a elementos del robot, elementos físicos o externos. En la planificación y navegación vamos a entrar en la parte cognitiva del sistema del robot.

Los aspectos específicos de cognición están fuertemente unidos a la movilidad y conseguir que el robot se mueva de forma segura y eficaz. La navegación de un robot involucra la planificación de una serie de acciones para la consecución de una meta. Cuando se van a ejecutar estas acciones, necesitamos un apartado reactivo dentro del robot, ya que en el camino planificado se puede encontrar con obstáculos que el robot debe saber evitar. Por lo tanto a la hora de realizar una buena navegación se hace estrictamente necesario una buena planificación y un buen sistema reactivo.

PLANIFICACIÓN DE RUTA

Cuando planificamos una ruta se intenta dar una completa descripción de la geometría de un robot y un entorno estático que puede tener o no obstáculos. Nuestra meta es encontrar una ruta libre de obstáculos para que un robot se mueva desde una posición y orientación inicial hasta la posición y orientación final (Ge, y otros, 2006).

Aunque muchos algoritmos de planificación difieren entre ellos en múltiples detalles, todos siguen un patrón definido.

El primer paso es situar al robot en un punto de un nuevo mapa abstracto llamado configuración del espacio. Este mapeo transforma el problema original en un problema de planificación de ruta de movimiento a un punto. Lo siguiente que debemos realizar es la conectividad entre los puntos que se han incluido en el mapa abstracto. Y por ultimo debemos buscar en el mapa creado una ruta que nos lleve desde el origen hasta la meta.

Una importante consideración a tener en cuento cuando hablamos de algoritmos de planificación es la completitud. Un algoritmo es completo si nos devuelve una ruta valida siempre que esta exista o nos reporta que no existe ruta en el caso de que, efectivamente, no la haya (Ge, y otros, 2006).

Figura 14: patrón de planificación de rutas (Ge, y otros, 2006)

NAVEGACIÓN

La navegación es la combinación de la planificación de rutas, superación de obstáculos, localización e interpretación perceptual. Una manera de construir este objetivo podría ser el desarrollo de un diseño personalizado de una aplicación específica. Esta solución podría ser válida cuando se trata con robots sencillos pero si tratamos con robots avanzados, la solución descrita se queda corta (Siegwart, y otros, 2004).

Existen dos tipos de navegación que se pueden diferenciar, navegación global y navegación local. Como su nombre indica, la navegación global consiste en la recreación de rutas por las que el vehículo, robot o sistema debe pasar para llegar desde el punto A en el que se encuentra en este mismo hasta instante hasta el punto objetivo B pasando por una cantidad de puntos variables. Esta navegación no tiene en cuenta la posibilidad de encontrar posibles obstáculos o interrupciones, simplemente se encarga de suministrar la información acerca del camino óptimo para alcanzar la meta.

Este tipo de navegación es extremadamente eficaz cuando se está ante un entorno totalmente conocido, ya sea por mapa o por otro tipo de información. Si se conocen todos los elementos que pueden interferir en el camino de nuestro vehículo, el sistema de navegación global creará una ruta óptima evitando todo obstáculo que se conozca.

Por otro lado, la navegación local está enfocada al control en cada momento del vehículo. Esta navegación sí que tiene en cuenta los posibles obstáculos o cambios de ruta que se deban realizar de manera obligatoria ya que, la navegación local se encarga de mover el vehículo de manera continua siguiendo las indicaciones que la navegación global ha podido indicarle. También podría trabajar sin una navegación global, simplemente circulando de manera infinita y sin objetivo fijado evitando obstáculos.

La navegación local se suele utilizar o suele ser más efectiva cuando el entorno en el que se mueve el robot no es conocido, es decir, no existe ningún mapa sobre el que guiar los movimientos u obtener la posición de otros objetos u obstáculos. Además, para realizar un buen sistema de navegación local, y como se puede adivinar de todos los comentarios mencionados, es absolutamente imprescindible contar con sensores que nos proporcionen constantemente información de la situación del entorno.

Muchos sistemas poseen ambas navegaciones. Una global que indica el camino optimo hacia el punto objetivo y una local para que, una vez decidida la ruta, conduzca al robot de manera satisfactoria hasta dicha meta evitando obstáculos y demás elementos que puedan entorpecer el camino de dicho robot.

En lo que se refiere al proyecto, se utiliza una navegación local guiada mediante las líneas que delimitan la carretera. En este sentido, el problema propuesto es parecido a los robots que deben seguir una línea que pasa por debajo de ellos y que está pintada en el suelo haciendo formas y giros.

Por lo tanto, el trabajo que el robot realiza en este proyecto es comprobar la orientación de las líneas y su tendencia y en base a dichos datos obtener que patrón está siguiendo la carretera en ese instante. Para el proyecto se ha supuesto que no existen obstáculos, puesto que sería muy complicado probar un sistema de evasión de obstáculos dentro del entorno propuesto por RUC. La única restricción tomada en cuenta a la hora de navegar localmente es la de los semáforos, los cuales se comprueban siempre antes de tomar cualquier otra decisión. Una vez comprobado que el semáforo no nos impide el paso, se continúa con la gestión de los datos y la toma de decisiones.

Como se comenta en el apartado de futuras ampliaciones, sería posible realizar un sistema que utilizara el GPS proporcionado por RUC para la realización de una navegación global mediante la ayuda del GPS. Con dicha navegación el problema ya no consistiría solo en mover el vehículo, sino dirigirlo hacia una meta especificada por el camino óptimo, aunque sería necesario un reconocimiento inicial para dar a conocer el mapa al GPS puesto que dicho mapa es desconocido para el sistema.

2.7.- VISIÓN ARTIFICIAL

La visión artificial es una disciplina que engloba todos los procesos y elementos que dotan de ojos a una máquina. Para Nalwa (1993), “la visión artificial o comprensión de imágenes describe la deducción automática de la estructura y propiedades de un mundo tridimensional, posiblemente dinámico, bien a partir de una o varias imágenes bidimensionales de ese mundo”.

Como es evidente, el diseño de un sistema de visión artificial por computador intenta simular lo que una persona humana capta por su sentido de la vista. Es decir, reconocimiento de figuras, objetos, distancia hasta ellos, textura que lo conforma y todas las características que un humano deduce de un objeto con solo verlo.

La visión, de una manera simple y resumida, consiste en capturar imágenes y procesar el contenido que hay en ellas para obtener información. Para un ordenador, la parte de captación de imágenes ya está hecha. Tan solo debemos utilizar el hardware deseado para capturar imágenes (cámaras web, cámaras digitales, videocámaras,…) y una vez obtenidas estas imágenes, debemos realizar la parte de procesamiento de imágenes, aunque esta fase es una ardua tarea.

Para que el ordenador pueda procesar la información contenida en la imagen se debe facilitar el entendimiento de la misma. Esto se realiza mediante transformaciones en la imagen de manera que la información que nos interese predomine en la imagen por encima de la información que no es relevante para nosotros.

2.7.1.- ETAPAS DE UN PROCESO DE VISIÓN ARTIFICIAL

Para elaborar un buen sistema de visión artificial se deben cumplir una serie de etapas que ayudaran a llevar un desarrollo ordenado del mismo.

- Capturar imagen: el primer paso, como ya se ha comentado con anterioridad, es obtener la imagen mediante cualquier sistema hardware disponible. Incluso, como es el caso de nuestro proyecto, es posible que se esté en un entorno virtual y las imágenes obtenidas sean simuladas de dicho entorno virtual. Es decir, no es necesario tener una imagen real o un entorno real para aplicar visión artificial.

- Preprocesamiento: una vez capturada la imagen, se procede a su preprocesamiento, donde trataremos la imagen con filtros y distintas técnicas que nos permitirán aumentar el porcentaje de éxito a la hora de analizar la imagen más adelante.

- Segmentación: el siguiente paso a realizar es la segmentación. Básicamente, la segmentación se encarga de dividir la imagen en las partes u objetos que la constituyen. Generalmente, este proceso es uno de los más complejos que existen en visión artificial aparte de ser un proceso bastante importante, ya que si se realiza una buena segmentación se pueden ampliar en gran medida las probabilidades de éxito. Sin embargo, en caso de realizarse una mala segmentación es probable que nos veamos avocados al fracaso.

- Elección de representación: cuando el proceso de segmentación finaliza, obtenemos una imagen de datos que contiene la frontera de la región o los puntos de la misma. En este momento se deben convertir esos datos para que puedan ser interpretados por el ordenador. Para ello, se elige la representación que deseamos para traducir estos datos. Aunque esta elección no es suficiente para interpretar estos datos, es la primera parte del proceso total. Llegados a este punto es necesario elegir una de las dos representaciones por las que podemos optar, estas son:

    - Representación por frontera: Es posible segmentar una imagen en regiones a través de la detección de la frontera de cada región para las cuales existe un cambio significativo de atributos a ambos lados de la frontera. Esta representación es apropiada cuando el objetivo se centra en las características de la forma externa como concavidades, esquinas o convexidades.

    - Representación por regiones: Esta representación es la más simple de todas y consiste en agrupar los pixeles vecinos con amplitud similar para finalmente acabar formando una región. Esta representación es apropiada cuando el objetivo se centra en propiedades internas como la textura o el esqueleto

- Parametrización: este proceso se centra en la extracción de rasgos que nos den alguna información cuantitativa que sea interesante o rasgos que sean básicos a la hora de diferenciar objetos. Es la segunda parte de la interpretación de la imagen de datos que se ha obtenido de la fase de segmentación. Una vez completada la parametrización, obtendremos una imagen totalmente legible para el ordenador.

- Reconocimiento: proceso que asigna una etiqueta a los objetos que se hayan identificado.

- Interpretación: asigna el significado a todo el conjunto de objetos que se han detectado en la imagen.

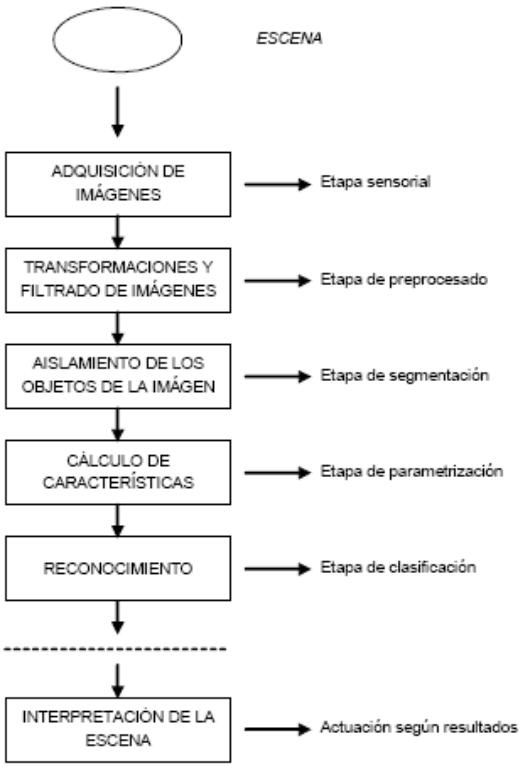


Figura 15: diagrama de las etapas de un sistema de visión artificial

Para el ámbito del proyecto, puede que algunas de estas fases sean omitidas o no se realicen ya que las imágenes que las cámaras que dispone el vehículo captan imágenes muy claras donde apenas existe ruido y otros elementos que interfieran en la identificación de objetos.

Figura 16: imagen capturada por una de las webcams del vehículo

Dichas imágenes son las que las cámaras simuladas se encargan de capturar del mundo virtual y por ello estás imágenes sintéticas carecen de algunos obstáculos que, de ser imágenes de un mundo real, podríamos encontrar. Hay que recordar que estamos hablando de un entorno virtual, no un entorno real donde existen multitud de factores que pueden interferir en la imagen.

2.7.2.- COMPONENTES DE UN SISTEMA DE VISIÓN ARTIFICIAL

Para la construcción de un sistema de visión artificial son necesarios los siguientes componentes hardware:

- Sensor óptico: el sensor puede ser una o varias cámaras a color o monocromo que produzca una imagen completa del dominio del problema en un intervalo determinado de tiempo.

- Tarjeta de adquisición de imagen: el único cometido que tiene esta tarjeta es digitalizar la imagen de manera que sea procesable por el ordenador.

- Ordenador: como es evidente, necesitamos un ordenador para almacenar la imagen y para albergar el software que se encargará de procesar la imagen. Hay casos en los que no hay un ordenador propiamente dicho, como en el caso de los robots, pero sí que contienen el software necesario para procesar la imagen y el soporte adecuado para almacenar esas imágenes.

- Monitor: para visualizar las imágenes y sus transformaciones es necesario un monitor que muestre dichas imágenes y su cambio tras cada proceso.

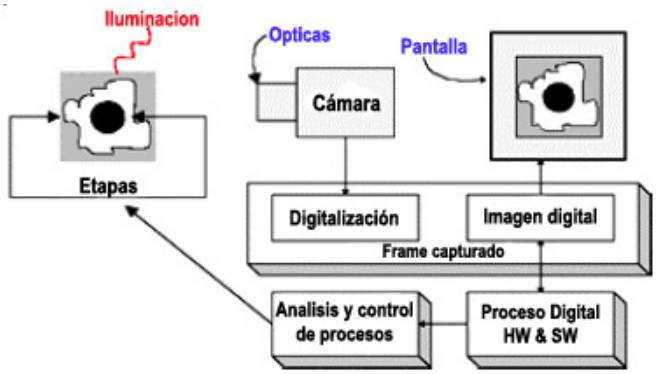


Figura 17: bloques de un sistema de visión artificial

Para el ámbito del proyecto, algunos de estos elementos no son necesarios. Por ejemplo, no disponemos de una tarjeta de adquisición de imagen ya que la imagen capturada por nuestras cámaras (sensores ópticos) ya están digitalizadas y en un formato que el ordenador puede comprender.

2.8.- HERRAMIENTAS UTILIZADAS

En este apartado se va a describir las aplicaciones y herramientas básicas en la realización del proyecto.

2.8.1.- MICROSOFT ROBOTICS DEVELOPER STUDIO

Microsoft Robotics Developer Studio es una herramienta de desarrollo que permite realizar aplicaciones robóticas basadas en servicios para varios dispositivos hardware. En el caso que nos atañe, no se utilizará ningún elemento hardware, ya que todo el entorno de este proyecto es simulado. Pero ello no es problema ya que MRDS posee un simulador mediante el cual logra suplir la necesidad de tener un hardware costoso con el que investigar.

Mediante los recursos disponibles en este entorno de desarrollo se logrará simular un robot y todos los componentes necesarios para hacerlos funcionar. Además, se podrá interactuar con todos los elementos simulados de dicho robot de manera semejante a como se haría si se dispusiera de un hardware real.

Esta herramienta está enfocada tanto para profesionales del sector con un nivel muy avanzado de conocimientos como para nuevos estudiantes que se quieran embarcar en el mundo de la robótica y no dispongan del hardware necesario para realizar una investigación concreta. En este sentido, MRDS es muy versátil y, una vez superado el periodo de aprendizaje, se pueden lograr grandes resultados sin que el problema sea una gran complicación.

Para lograr estos objetivos descritos, MRDS ofrece una serie de herramientas que nos facilitan el trabajo.

MRDS contiene servicios en tiempo de ejecución (*runtime*), gracias a que utiliza la biblioteca Concurrency and Cordination Runtime (CCR) con la cual es posible le realización de aplicaciones asíncronas. El CCR es la elección más correcta cuando tratamos problemas en los que cada componente se comunica mediante mensajes asíncronos, como es el caso típico en el flujo de datos de sensores y actuadores de un robot. El CCR se encarga de gestionar estos mensajes y que todos lleguen a su destino en el momento que es necesario y de manera correcta.

2.8.2.- MICROSOFT VISUAL PROGRAMMING LENGUAGE

Microsoft Visual Programming Lenguage (MVPL) es una herramienta de programación visual que se encuentra dentro del paquete de MRDS y que nos permite interactuar con los servicios de manera sencilla y visual. En vez de programar de la manera habitual, línea tras línea, MVPL propone otra manera, distinta y revolucionaria que es la de crear un gráfico en la que los servicios se comunican entre sí mediante interacciones (líneas). Con estas interacciones cada servicio manda la información recogida o necesaria a otro servicio.

Esta herramienta, aunque revolucionaria, una vez que sobrepasas un cierto nivel de exigencia, pierde utilidad ya que se hace necesario un control más exhaustivo de todo el proyecto, control que únicamente ofrece el código.

Su utilización dentro de este proyecto se ha limitado a razones de aprendizaje y de realización de pruebas a nivel básico para lograr un conocimiento en lo que se refiere a servicios ya que al ser completamente visual es más fácilmente asimilable para una persona sin conocimientos en este campo.

2.8.3.- MICROSOFT VISUAL SIMULATION ENVIRONMENT

MRDS tiene como objetivo el llegar a una gran cantidad de usuarios, tanto profesionales y expertos como personas que desean iniciarse en el mundo de la robótica. Uno de los grandes aspectos que contribuyen a esta adaptación tan grande es Microsoft Visual Simulation Environment (MVSL).

El MVSL es un entorno de simulación, es decir, como si de un juego se tratara, en MRDS es posible simular cualquier tipo de entorno en tres dimensiones, desde ciudades, casas hasta lugares imaginarios o ficticios, todo ello con las reglas físicas que el usuario desee.

Como se ha comentado con anterioridad, MVSL está diseñado de tal manera que satisface las exigencias de los usuarios más avanzados y, al mismo tiempo, permite al usuario con poca experiencia de programación acceder de una manera sencilla a un entorno virtual. La integración de Ageia PhysX[2] permite la recreación de físicas avanzadas, logrando un mundo más real gracias a la integración de elementos como la gravedad, rozamiento, flotabilidad, etc.

[2] Ageia PhysX, motor de física: http://www.nvidia.es/object/physx_accelerator_es.html

2.8.4.- MICROSOFT VISUAL STUDIO

Microsoft Visual Studio es un conjunto de herramientas de desarrollo diseñadas para facilitar a los diseñadores de software el enfrentamiento a problemas complejos con la máxima comodidad posible. Esta herramienta se trata de un software IDE (*Interface Development Enviroment*) y nos proporciona un editor de código, un compilador con el que compilar nuestro código y una herramienta de depuración de código entre otras características, las cuales le hacen ser.

Además de dichas características principales que hacen que MSVS sea un software IDE, posee una alta compatibilidad con Windows y facilidad de manejo en dicho sistema operativo, herramientas de desarrollo avanzadas, funcionalidad con bases de datos y otra serie de características que ayudan al programador y facilitan su complicado trabajo.

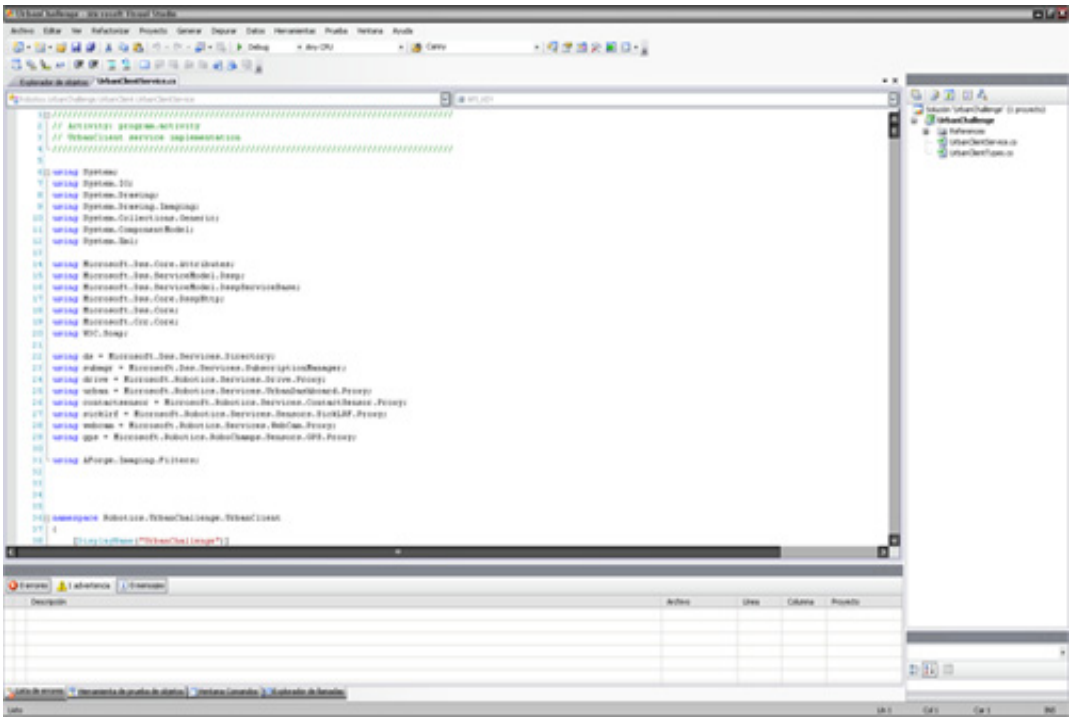


Figura 18: ventana del software IDE Microsoft Visual Studio

MSVS soporta lenguajes como Visual Basic, Visual C++, Visual C# y Visual J# y dichos lenguajes aprovechan las funciones de .NET Framework que ofrece acceso a tecnologías clave para simplificar el desarrollo de aplicaciones Web ASP y Servicios Web XML. Además también soporta HTML, XHTML, Javascript y CSS, lenguajes básicos a la hora de realizar páginas web.

Para el proyecto, únicamente se utilizará el lenguaje C# a pesar de que MSVS ofrece compatibilidad con múltiples lenguajes.

2.8.5.- .NET FRAMEWORK

.NET Framework es un componente integral de Windows que admite la creación y la ejecución de la siguiente generación de aplicaciones y servicios Web XML. El diseño de .NET Framework está enfocado a cumplir los objetivos siguientes:

- Proporcionar un entorno coherente de programación orientada a objetos, en el que el código de los objetos se pueda almacenar y ejecutar de forma local, ejecutar de forma local pero distribuida en Internet o ejecutar de forma remota.

- Proporcionar un entorno de ejecución de código que reduzca lo máximo posible la implementación de software y los conflictos de versiones.

- Ofrecer un entorno de ejecución de código que fomente la ejecución segura del mismo, incluso del creado por terceras personas desconocidas o que no son de plena confianza.

- Proporcionar un entorno de ejecución de código que elimine los problemas de rendimiento de los entornos en los que se utilizan secuencias de comandos o intérpretes de comandos.

- Ofrecer al programador una experiencia coherente entre tipos de aplicaciones muy diferentes, como las basadas en Windows o en el Web.

- Basar toda la comunicación en estándares del sector para asegurar que el código de .NET Framework se puede integrar con otros tipos de código.

.NET Framework contiene dos componentes principales: *Common Language Runtime* y la biblioteca de clases de .NET Framework.

Common Language Runtime es el fundamento de .NET Framework. El motor en tiempo de ejecución se puede considerar como un agente que administra el código en tiempo de ejecución y proporciona servicios centrales, como la administración de memoria, la administración de subprocesos y la interacción remota, al tiempo que aplica una seguridad estricta a los tipos y otras formas de especificación del código que fomentan su seguridad y solidez. De hecho, el concepto de administración de código es un principio básico del motor en tiempo de ejecución. El código destinado al motor en tiempo de ejecución se denomina código administrado, a diferencia del resto de código, que se conoce como código no administrado.

La biblioteca de clases, el otro componente principal de .NET Framework, es una completa colección orientada a objetos de tipos reutilizables que se pueden emplear para desarrollar aplicaciones que abarcan desde las tradicionales herramientas de interfaz gráfica de usuario (GUI) o de línea de comandos hasta las aplicaciones basadas en las innovaciones más recientes proporcionadas por ASP.NET, como los formularios Web Forms y los servicios Web XML.

2.8.6.- MICROSOFT VISUAL C# 2.0

El lenguaje en el que se realizará el proyecto es Microsoft Visual C# 2.0. Dicho lenguaje se creó para la implementación de multitud de aplicaciones que se ejecutan en el entorno .NET Framework. Este lenguaje es simple, eficaz, con seguridad de tipos y orientado a objetos. Por supuesto, MSVS soporta el lenguaje C# con lo que, utilizando MSVS como editor, obtienes se obtiene una gran cantidad de ventajas, como plantillas, facilidad a la hora de depurar y asistentes para código, entre otras.

La sintaxis de C# es muy expresiva, pero también es sencilla y fácil de aprender. La sintaxis de C# basada en signos de llave podrá ser reconocida inmediatamente por cualquier persona familiarizada con C, C++ o Java ya que reutiliza muchos aspectos del código C.

La sintaxis de C# simplifica muchas de las complejidades de C++ y proporciona características eficaces tales como tipos de valores que admiten valores NULL, enumeraciones, delegados, expresiones lambda y acceso directo a memoria. C# admite métodos y tipos genéricos, que proporcionan mayor rendimiento y seguridad de tipos, e iteradores, que permiten a los implementadores de clases de colección definir

comportamientos de iteración personalizados que el código cliente puede utilizar fácilmente. Como lenguaje orientado a objetos, C# admite los conceptos de encapsulación, herencia y polimorfismo.

2.8.7.- AFORGE.NET FRAMEWORK

Para el tratamiento de las imágenes se ha utilizado una librería externa la cual permite hacer multitud de operaciones con las imágenes. Operaciones que, de haber sido realizadas en el proyecto a mano hubieran traído consigo una carga considerable de trabajo y que no forma parte del objetivo principal del proyecto.

El autor de esta librería es Andrew Kirillov y nos permite realizar toda clase de operaciones, desde un sencillo cambio de color a blanco y negro, hasta las más avanzadas técnicas de detección de bordes de una manera sencilla y rápida. La versión utilizada es la 1.7.0 ya que es la última versión estable y se puede conseguir desde su página oficial[3], donde además se puede obtener información variada acerca de la librería y del autor. La librería está desarrollada en C# y con el objetivo de ser utilizada en ese mismo lenguaje.

El contenido de AForge.NET no se limita al tratamiento de imágenes, sino que es un conjunto de librerías y programas de ejemplo que muestran el potencial del *framework*:

- AForge.Imaging: librería con rutinas y filtros de procesamiento de imágenes.
- AForge.Vision: librería de visión artificial.
- AForge.Neuro: librería de redes neuronales.
- AForge.Genetic: librería de algoritmos genéticos.
- AForge.MachineLearning: librería de aprendizaje máquina.
- AForge.Robotics: librería que ofrece soporte a varios kits robóticos.
- AForge.Video: conjunto de librerías para el procesamiento de video.

Como se puede observar, AForge contiene mucho más que tratamiento de imágenes pero para el objetivo de este proyecto solo se utilizará *AForge.Imaging*. El resto de contenido de la librería no se utilizará.

[3] Página oficial AForge.NET: http://www.aforgenet.com/framework/

3.- GESTIÓN DEL PROYECTO

3.1.- CICLO DE VIDA DEL SOFTWARE

El ciclo de vida de un software cualquiera es, según podemos encontrar en la norma *The Institute of Electrical and Electronics Engineers* (IEEE) 1074, como "aproximación lógica a la adquisición, el suministro, el desarrollo, la explotación y el mantenimiento del software". Mientras que la norma International Organization for Standardization (ISO) 12207-1 entiende un ciclo de vida del software como "un marco de referencia que contiene los procesos, las actividades y las tareas involucradas en el desarrollo, la explotación y el mantenimiento de un producto software, abarcando la vida del sistema desde la definición de los requisitos hasta la finalización de su uso".

Ambas consideran una actividad como un conjunto de tareas y una tarea como una acción que transforma entradas en salidas.

Introducido lo que un ciclo de vida significa, existen varios modelos de ciclo de vida. Cada uno posee sus ventajas y desventajas frente a otro ciclo de vida. Ofrecer una explicación detallada de las diferencias entre dichos ciclos y sus ventajas e inconvenientes respecto a otros se sale de lo que este proyecto abarca. Simplemente, se definirá el ciclo de vida elegido para la realización de este proyecto y por qué se ha seguido dicho ciclo y de qué manera ayuda este ciclo de vida a la realización del software propuesto.

Por otro lado, algunas tareas propias del ciclo de vida no serán realizadas, como es el mantenimiento del software ya que no se ofrecerá ningún tipo de soporte posterior a la entrega del proyecto por razones obvias, puesto que el propósito del mismo no es realizar un software completamente funcional sino demostrar los conocimientos adquiridos durante los cursos realizados y aplicar dichos conocimientos demostrando la capacidad del alumno.

3.1.1.- PROCESOS DEL CICLO DE VIDA DEL SOFTWARE

Siguiendo la norma ISO 12207-1, las actividades que se pueden realizar se dividen en cinco procesos principales, ocho procesos de soporte y cuatro procesos generales tal y como se muestra de una manera más gráfica en la figura 19.

Estos procesos no obligan a elegir ningún modelo de ciclo de vida de los existentes en la actualidad ni prescribe como realizar ninguno de los procesos mostrados en la figura. Simplemente muestra una organización posible a la hora de realizar un software completo. Así mismo, estos procesos son los recomendados o mostrados por la norma ISO 12207-1. La norma IEEE 1074 tiene sus propios procesos y su propia organización que puede diferir o no de la mostrada en la norma ISO 12207-1.

En el ámbito del proyecto habrá procesos que no se realizaran o que se saltarán. Procesos como adquisición, suministro, formación, mejora… no serán realizados en este software puesto que se salen del propósito principal de este proyecto. Por ello, y para no desviarnos de nuestro ámbito se comentaran brevemente los procesos que se realizarán en este proyecto.

PROCESO DE DESARROLLO

El proceso de desarrollo será el proceso vital. Es el proceso en el cual nos basaremos para la realización del proyecto y el cual contiene las tareas básicas a la hora de realizar un software. No obstante, es el proceso más importante, no solo para este proyecto, sino para cualquier proyecto, por ello, se detallaran de una manera más profunda las fases que contiene este proceso.

- Análisis de requisitos del sistema: se especifican los requisitos del sistema, incluyendo las funciones y las capacidades que debe incluir, los requisitos de seguridad, de interacción hombre-máquina, de

interfaces, de operaciones y de mantenimiento, las restricciones aplicables al diseño y los requisitos para su aceptación.

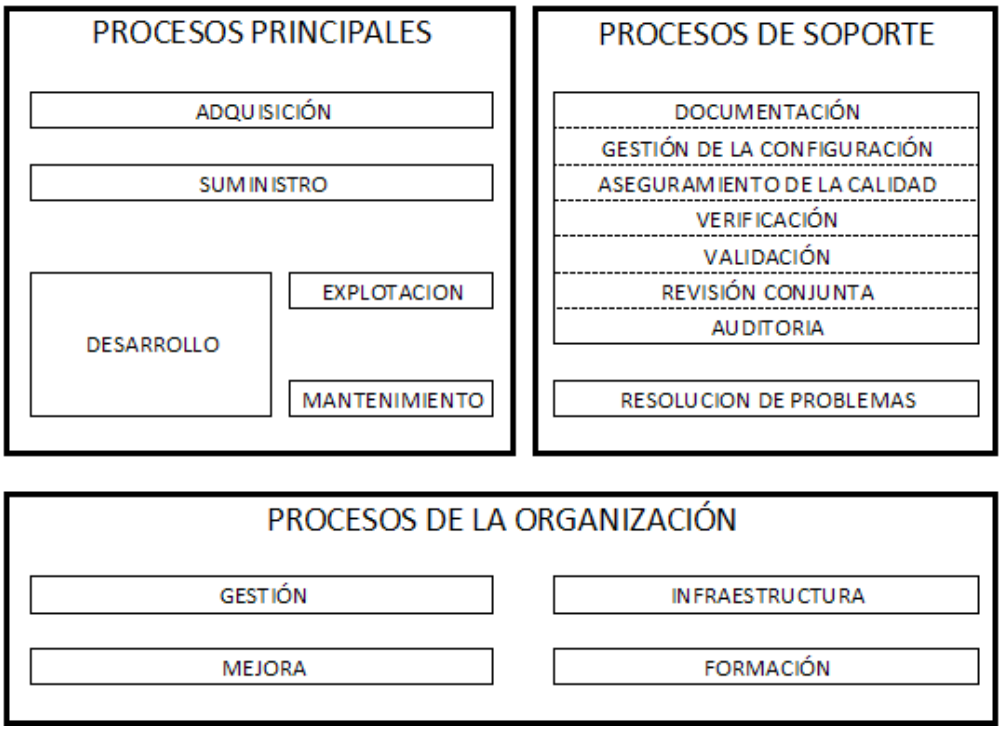


Figura 19: procesos del ciclo de vida software según ISO 12207-1

- Diseño de la arquitectura del sistema: se establece la arquitectura de alto nivel, la cual identificará los principales componentes de hardware y de software, así como las operaciones manuales del sistema.
- Análisis de los requisitos del software: se establecen y documentan dichos requisitos, incluyendo la especificación de las características de calidad que debe cumplir el sistema.
- Diseño de la arquitectura del software: el desarrollador debe transformar los requisitos del software en una arquitectura o estructura de alto nivel que identifica sus componentes principales.
- Diseño detallado del software: se realiza un diseño detallado para cada componente software, incluidas las bases de datos.
- Codificación y pruebas del software: se desarrollan y se documentan los distintos componentes software y las bases de datos. Posteriormente se prueba cada uno de ellos para asegurar que satisfacen los requisitos.
- Integración del software: se integran los componentes del software y se prueban según sea necesario.
- Prueba del software: el desarrollador lleva a cabo la prueba de cualificación en función de los requisitos especificados.
- Integración del sistema: se integran los elementos software y hardware junto con las operaciones manuales.
- Prueba del sistema: análoga a la de software, pero se llevará a cabo según los requisitos del sistema definidos con anterioridad.
- Instalación del software en el entorno de explotación final donde vaya a funcionar.
- Soporte del proceso de aceptación del software: el desarrollador debe dar su apoyo a la revisión y aceptación y a la prueba del software por parte del comprador.

Al igual que el resto de procesos, las tareas descritas en el proceso de desarrollo no se realizaran íntegramente debido al tiempo limitad del que se dispone para la realización de la misma y por el volumen.

Algunas tareas si se realizaran íntegramente pero otras pueden realizarse íntegramente, en parte o incluso no realizarse.

3.1.2.- MODELO EN ESPIRAL

El modelo de ciclo de vida que va a ser utilizado para la realización del proyecto es el modelo en espiral creado por Boehm en 1988 dando solución a los principales inconvenientes que posee el modelo en cascada, modelo imperfecto para determinadas situaciones. El modelo en espiral consta de una serie de fases que se repiten en forma de ciclo siempre de la misma manera y con la misma metodología.

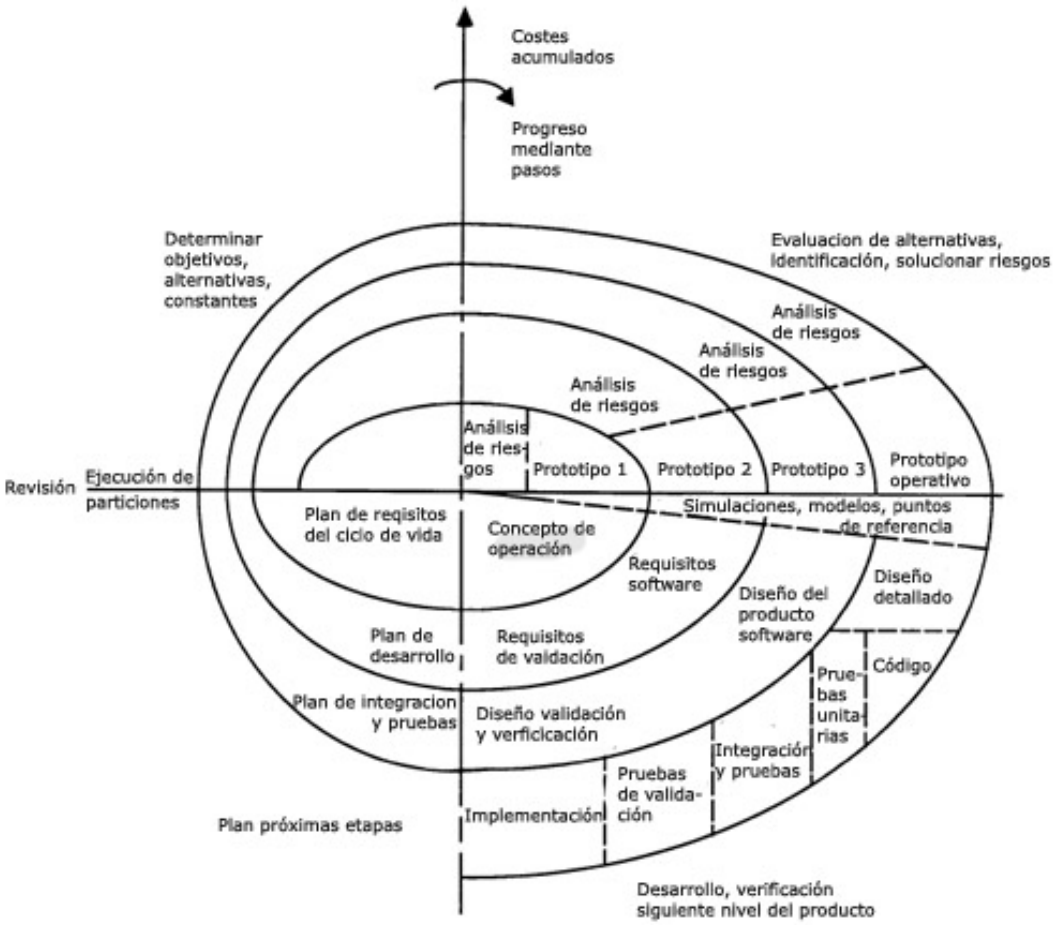


Figura 20: modelo en espiral de Boehm (Boehm, 1988)

Para entender mejor el funcionamiento de este modelo de ciclo de vida se explicara en qué consiste de manera básica y resumida cada una de las fases que se repiten en un ciclo completo.

IDENTIFICACIÓN

Cada ciclo de la espiral comienza con identificación de los objetivos de la parte del producto que está siendo elaborada, las alternativas principales de la implementación de esta porción del producto y por último las restricciones impuestas para cada alternativa.

En el apartado de alternativas principales, dichas alternativas pueden ser actividades no relacionadas directamente con el software aunque sí con el proyecto que atañe, como por ejemplo alguna tarea relacionada con la gestión o de personal.

EVALUACIÓN

En este paso, se evalúan las diferentes alternativas que se plantean teniendo en cuenta los objetivos a conseguir y las restricciones impuestas. Frecuentemente, este paso identifica las áreas de incertidumbre del proyecto con sus correspondientes riesgos. En el caso de la existencia de riesgos, se pasaría a una *subfase* la cual conlleva la formulación de una estrategia efectiva en coste para resolver dichos riesgos. Dicha estrategia se podrá apoyar en cualquier tipo de fuente de información a la que tengamos acceso como la utilización de prototipos, encuestas, bancos de pruebas, etc.

DESARROLLO

En este paso se desarrolla el producto o productos que se han estimado para esta iteración. Una vez desarrollados se verifican los riesgos que se han estudiado en el punto anterior y si el producto final cumple con los requisitos que se solicitan. Una vez realizadas las pruebas necesarias para verificar dicho desarrollo se pasaría a la última fase de este modelo de ciclo de vida.

PLANIFICACIÓN

Se lleva a cabo la planificación de la siguiente iteración en el ciclo de vida y se prepara para volver a empezar de nuevo con el ciclo.

Cada ciclo, finalmente, se completa con una revisión en la que participan las principales personas u organizaciones que han tenido o tienen relación con el producto.

Este proceso se repetiría las veces que fuera necesario hasta alcanzar un producto final completo que satisfaga todos los requisitos propuesto por el cliente.

Este modelo reconoce explícitamente las diferentes alternativas para alcanzar los objetivos de un proyecto, identifica los riesgos asociados con cada una de las alternativas y las diferentes maneras de resolverlos, divide el proyecto en ciclos, lo que implica que existe un acuerdo para los cambios a realizar o para terminar dicho proyecto en función de lo aprendido en su desarrollo y se adapta a cualquier tipo de actividad incluso en algunas en las que no existen otros métodos.

3.2.- ALCANCE DEL PROYECTO

El trabajo realizado no sólo se ha limitado a la creación del sistema autónomo capaz de mover un vehículo sino que ha sido necesario realizar un trabajo previo de adaptación al entorno, búsqueda de datos e información.

Posteriormente, se puede observar la planificación creada para el proyecto pero previo a esa planificación fue necesario estudiar el entorno sobre el que se estaba trabajando. Gran parte del tiempo inicial se dedicó a estudiar el lenguaje en el cual se solicitaba el proyecto, fue necesario estudiar con detalle y mediante ejemplos útiles las nuevas tecnologías aportadas por MRDS.

En esos primeros pasos, prácticamente la totalidad del tiempo se dedicó a la comprensión de las herramientas de trabajo y a la creación de pequeños ejemplos que aumentasen en complejidad paulatinamente y a su vez que se encontraran nuevos desafíos y problemas que, al final del proyecto pueden parecer sencillos, pero que para un aprendiz pueden resultar bastante complejos.

Una vez estudiado las herramientas con las que se iba a trabajar, era necesario conocer qué ofrecía RUC, qué nos proporcionaba y qué no. Es de vital importancia conocer el entorno donde se va a enmarcar el proyecto y saber hasta dónde puede llegar el desarrollo los límites y las libertades que ofrece trabajar sobre una plataforma. En este aspecto, es de gran ayuda que RUC se pueda ejecutar sin

automatismos, es decir, poder controlar de manera manual el vehículo. De esta manera los pequeños cambios que se puedan introducir eran resaltables.

Todo este trabajo de adaptación, aunque no está reflejado en ninguna planificación ni parece formar parte del proyecto, está ahí y es vital para el desarrollo final del proyecto.

Una vez finalizado, o en otras palabras, una vez que ya se ha terminado el apartado de "formación", se comenzó con lo que es el proyecto propiamente dicho. A partir de este momento se intentó seguir una estructura de prototipos para ceñirse al ciclo de vida elegido lo máximo posible. Para ello, se procuró ir realizando pequeñas tareas y una vez realizadas, obtener información valiosa mediante las pruebas a las que se sometía ese prototipo. Con el resultado de esas pruebas, obtener los detalles, tanto los aciertos como los errores y tomarlo de base para el siguiente prototipo y así continuamente hasta que se alcanzase un prototipo que cumpliera lo máximo posible con los objetivos marcados.

Como se comentará en apartados posteriores, la realización del proyecto se puede dividir en dos fases claramente diferenciadas, la primera parte donde los prototipos seguían un enfoque reactivo y la segunda parte o parte final en la que se implantó el paradigma híbrido como solución final al problema abordado.

Se han realizado cinco prototipos que siguen el enfoque reactivo y dos prototipos que siguen el enfoque híbrido. Los cinco prototipos creados en torno al enfoque reactivo son debidos a que los primeros de ellos sirvieron de toma de contacto, para saber cómo reaccionaba el sistema, como ofrecía los datos, etc. Finalmente el quinto prototipo mostró de manera clara la necesidad un cambio importante de filosofía.

A pesar de la diferencia en el número de prototipos entre ambos enfoques, el tiempo que se utilizó para cada uno de los enfoques es similar. De manera muy general, la primera etapa se utilizó para desarrollar el sistema de análisis de imágenes y obtención de información de las mismas, aunque también se procuro incrementar la efectividad en el movimiento del vehículo. En la segunda etapa, la etapa híbrida, se enfocó el desarrollo en el movimiento del vehículo ya que el sistema de gestión de las imágenes ya estaba desarrollado y ofreciendo datos válidos por lo que fue más fácil centrarse en él.

3.3.- PLANIFICACIÓN

La planificación que se ha realizado para el proyecto es una planificación en la que se ha enfatizado en las pruebas, ya que es la vital fuente de información de lo que se dispone. Aunque no se puede descuidar el resto de aspectos que hacen que el software vaya por buen camino. En este sentido, se ha intentado realizar cada fase lo más fiel posible para que cada prototipo sea una avance y no un paso atrás.

Como se comenta, se han realizado siete prototipos, cinco reactivos y dos híbridos con las principales fases que recomienda el ciclo de vida, como son análisis, diseño, implementación y pruebas y una vez terminado este ciclo comienza otro nuevo para el siguiente prototipo basado en el anterior. De esta manera, agregando elementos al proyecto se alcanza la solución final de manera casi involuntaria. Así, el proyecto se hace más llevadero y es más fácil trabajar localizando pequeñas metas que trabajar en busca de una meta final muy amplia que podría llevar a confusión.

La planificación, por motivos de limpieza y síntesis, se ha llevado al "Anexo B: planificación del proyecto" ya que es un diagrama de Gantt bastante amplio y se ha preferido apartarlo allí para que se pueda observar en cualquier momento y no ensucie este apartado del documento.

4.- TRABAJO REALIZADO

4.1.- ANÁLISIS

En este apartado se describirá el proyecto realizado. En este apartado se refleja de manera formal todo lo que el proyecto contiene para alcanzar el éxito del producto y que sea un producto completo y que alcance los objetivos definidos.

4.1.1.- PERSPECTIVA DEL PRODUCTO

El producto desarrollado en este proyecto es un sistema automático de control de un vehículo mediante la utilización de cámaras, laser de aproximación, sensor de impacto y GPS, como se describió en los objetivos de la introducción.

El producto cubre una serie de necesidades:

- El sistema autónomo conduce un vehículo estándar de manera segura siguiendo la vía por la que se encuentra y respetando las normas de tráfico asignadas a la vía.

- Se hace uso de las cámaras que el vehículo posee para la realización de un sistema de visión artificial con el que poder entender que sucede en el exterior y poder decidir qué hacer en cada momento.

- Para el tratamiento de las imágenes obtenidas por las cámaras, se hace uso de una librería de visión artificial con licencia libre y uso permitido que ayudará en la realización de tareas complejas que no entran dentro del ámbito de producto.

- El producto incorpora un sistema de toma de decisiones en base a la información obtenida por las cámaras del vehículo y los láseres de aproximación.

- El producto también incorpora un sistema de ejecución de acciones que indica lo que el vehículo hará y como se comportará.

- Se hace uso de las metodologías estudiadas en el campo de la robótica y más concretamente, la utilización de la arquitectura reactiva y la arquitectura deliberativa formando un sistema que conforme una arquitectura híbrida que ayude a cumplir los objetivos y metas marcadas en cuanto a efectividad se refiere.

- El sistema se realiza en el marco de la competición RUC, el cual proporciona el entorno virtual y los elementos (cámaras, láser, sensor de impacto y GPS) necesarios para la realización del sistema autónomo.

- Se dispone de un sistema de control de extravíos que permitirá volver a la carretera en caso de que el vehículo se haya salido de la misma u orientarse para volver a obtener datos fiables que sitúen al vehículo en el mundo real.

- El proyecto se desarrolla en un entorno virtual donde un posible accidente no aumenta el coste del proyecto ni es necesario arriesgar la integridad del vehículo ni del mundo que le rodea.

- Aunque el entorno sea virtual este proyecto está preparado para integrarse en cualquier vehículo real siempre y cuando disponga de los mismos elementos hardware.

- El proyecto será lo más flexible en cuanto a reacciones con el entorno para que el vehículo se adapte a cualquier tipo de vía y distintos tipos de desafíos.

- El software desarrollado es modular, dando la posibilidad de realizar pequeños cambios de manera sencilla.

4.1.2.- CARACTERÍSTICAS DEL USUARIO

En este proyecto no es necesaria la interacción de ningún usuario. Tan solo se ha de poner en funcionamiento el sistema y una vez encendido, dicho sistema actúa de manera independiente hasta que de nuevo sea apagado.

Todos los elementos virtuales proporcionados deben cumplir su cometido ya que si dejara de prestar servicio alguno de ellos, el éxito en el funcionamiento del sistema se vería comprometido.

4.1.3.- RESTRICCIONES

El proyecto, debido a su gran amplitud posee una serie de restricciones para que, en el tiempo estimado, fuera posible su realización:

- Solo se trabaja con los elementos proporcionados por RUC en su kit de iniciación proporcionado para entrar en concurso.

- No se utiliza el elemento GPS proporcionado por RUC en su sistema virtual con lo que el vehículo repite trayectos aunque ya haya pasado por ellos.

- Por convenio general, en las rotondas y cruces se toma la primera salida a la derecha en caso de que exista en un cruce. Esta es una restricción derivada de no usar el GPS.

- La velocidad es constante ya estemos en curva o en recta.

- Las pruebas, al no disponer de más mapas, se realizan sobre el mapa base propuesto por RUC.

- El uso de una librería para el tratamiento de la imagen podría ser una restricción, aunque para el objetivo de este proyecto dicha librería cumple su función a la perfección.

4.1.4.- ENTORNO OPERATIVO

Para poder ejecutar este producto en un entorno virtual es necesaria una fuerte cantidad de requisitos debido a que consume gran cantidad de recursos del ordenador donde se ubique.

La restricción más importante en este tema se encuentra en el hardware gráfico el cual debe tener una potencia considerable siendo compatible con DirectX 9.0c[4] y, si es posible, con la tecnología Ageia PhysX. Además el consumo de memoria RAM también es considerable por lo que se recomienda tener mínimo un gigabyte de memora RAM. Finalmente la CPU debe ir en concordancia con el chip grafico y con la RAM que posee el ordenador para que todo el equipo tenga el equilibrio perfecto para ejecutar el programa.

En cambio, si el sistema se va a implementar sobre un sistema hardware físico, es decir, donde ya se disponga del vehículo físicamente, de las cámaras y demás elementos necesarios, la exigencia de un hardware gráfico queda eliminada y el producto se limita a trabajar con los elementos físicos proporcionados.

[4] DirectX 9.0c: http://www.microsoft.com/downloads/details.aspx?FamilyID=9226a611-62fe-4f61-aba1-914185249413&DisplayLang=es#Overview

4.2.- DISEÑO

En este apartado se explica cómo se pretende dar solución a todas las necesidades que se han planteado en el apartado de análisis. Para ello, se intenta aportar todo tipo de recursos tales como diagramas e imágenes para ver de una manera clara y concisa el funcionamiento del sistema.

Hasta ahora hemos delimitado el qué vamos a hacer, qué se desea conseguir con este proyecto. A partir de este punto, se explica cómo se va a realizar el proyecto para la consecución de las metas.

Para intentar comprender mejor el funcionamiento del proyecto se intenta dividir el mismo en módulos separados de la totalidad con el fin de entender individualmente el funcionamiento de cada módulo, una vez hecho esto, se conectan los distintos módulos que dan forma a la totalidad del proyecto.

De una manera genérica, se podría decir que el proyecto está formado por varios módulos que al final interaccionan entre sí para conseguir completar su tarea. Estos módulos son:

- El vehículo: El vehículo es el que contiene todos los elementos virtuales que nos prestaran la información para poder conducirlo de manera correcta. Sin estos elementos sería imposible la realización del mismo.

- Sistema de gestión de imágenes: este sistema es el encargado de gestionar las imágenes que capturan las webcams. Desde la solicitud del *frame* que en ese instante está gestionando la cámara hasta guardar una muestra válida con la información necesaria para la toma de decisiones. En ese proceso también se incluye el tratamiento de la imagen de manera que ayude a obtener los datos que nos interesan de una manera más clara, el proceso de búsqueda de esos datos que son de interés y la obtención de datos derivados de los obtenidos en la gestión anterior.

- Sistema de toma de decisiones: Finalmente, este sistema es el encargado de decidir con todos los datos obtenidos en el anterior sistema qué acción debe realizar el vehículo. Además de decidir qué acción debe realizar la ejecuta enviando la orden al dispositivo adecuado.

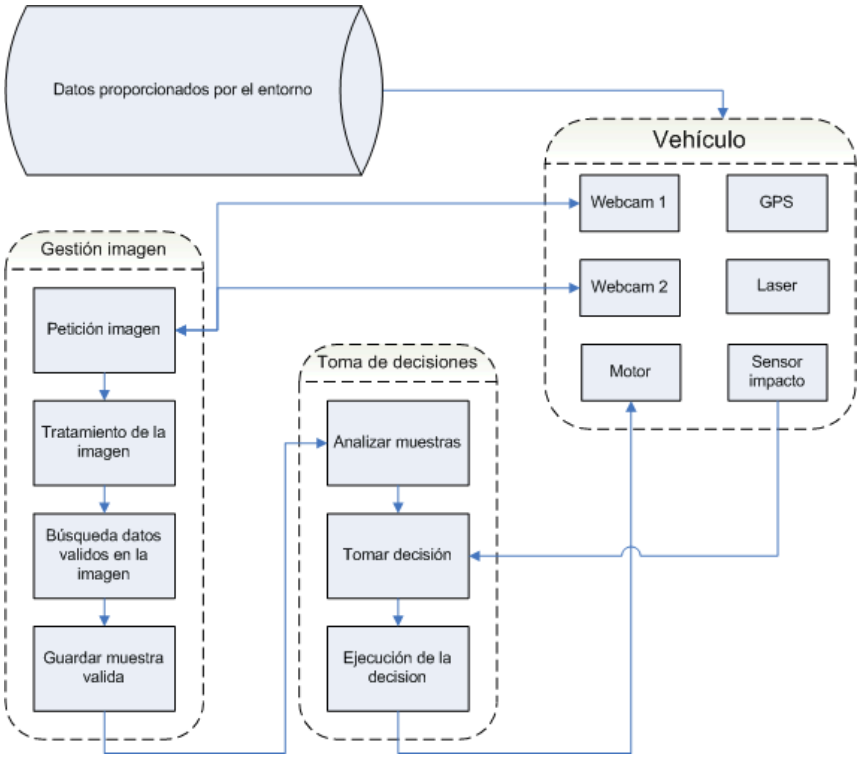


Figura 21: esquema de funcionamiento del sistema final

4.2.1.- MÓDULOS

MÓDULO VEHÍCULO

- Entrada: la entrada de este módulo es simplemente los datos proporcionados por el mundo real. En este caso, las imágenes que captan las webcams, la distancia que captura el laser de aproximación y la posición en la que está el coche el GPS. Estos sistemas, son los proporcionados por el kit de iniciación de RUC y no han sido modificados en absoluto, tan solo utilizados según las necesidades del proyecto.

  Además de los datos del mundo real, continuamente se están enviando datos al motor para que realice las acciones necesarias. El motor es el elemento con el que se comunica el sistema para realizar correcciones y para llevar el coche por donde el módulo de toma de decisiones indica.

- Proceso: el proceso que sigue este módulo es totalmente transparente para el desarrollador, no se sabe con certeza cómo calcula la posición el GPS o la distancia el laser de aproximación, tan solo se sabe que nos proporciona los datos que se necesitan y que son vitales. A pesar de considerarlo un módulo, es un módulo ya realizado y que, aunque es vital, no es importante cómo realiza su función, solo importa que la realice constantemente y siempre que sea solicitado por otros módulos. Una importante consideración a tener en cuenta es cómo se entiende el motor en el sistema dado por RUC.

  Aunque ya se comentó en la introducción de este documente es importante insistir en que los robots autónomos móviles soportados por el motor genérico de MRDS, el motor proporcionado por RUC, no posee dirección, ni ruedas directrices, ambas ruedas constantemente están dirigidas hacia el centro por lo que el giro se realiza aplicando potencia directamente a cada una de las ruedas por separado. Por ejemplo, en el caso de aplicar una fuerza de 0,2 en la rueda derecha y de 0,3 en la rueda izquierda, el vehículo giraría a la derecha debido a que la rueda izquierda ejerce más potencia. Aunque es obvio que el funcionamiento de un vehículo real no es así, lo que realmente estamos

manejando es un robot autónomo de un eje motriz aunque en la pantalla veamos un vehículo normal y corriente con sus dos ejes y un eje directriz.

En el mundo virtual, se puede observar un vehículo con un eje directriz e incluso ver como las ruedas, efectivamente, están giradas pero internamente lo que se hace es que se aplican diferentes potencias. Esta solución se ha hecho de manera estética para no ver el vehículo desplazarse lateralmente de manera antinatural.

Respecto a las webcams, cada una de ellas obtiene una perspectiva distinta lo que da juego a distintas posibilidades a la hora de tratar con las imágenes que obtienen. Al ser un entorno virtual, el trabajo a la hora tratar la imagen se facilita ya que en este mundo no se obtiene tanto ruido y tanta suciedad como podría pasar en un mundo real. Las imágenes son muchas más nítidas lo que nos permite actuar de una manera más restrictiva teniendo en cuenta menos variables

- Salida: las salidas de este proceso son varias. Las más útiles de cara a nuestro proyecto son las imágenes obtenidas por ambas cámaras web, la medición de aproximación del laser y el sensor de impacto en cada uno de los laterales del coche.

MÓDULO GESTIÓN DE IMÁGENES

- Entrada: Este módulo tiene como entrada las imágenes capturadas por las webcams que están situadas en el vehículo. Dichas imágenes son solicitadas por este módulo para su tratamiento. El lanzamiento de estas peticiones no se realiza aleatoriamente, sino que se controla con un reloj interno.

- Proceso: en resumen, como se ha comentado anteriormente, este módulo se encarga de tratar la imagen, buscar los datos interesantes para el modulo de toma de decisiones y guardar dichos datos.

Pero más concretamente, una vez que se recibe la imagen de la webcam, comienza su tratamiento. En un primer momento se busca por la imagen la posible existencia de semáforos. Esto se realiza con la imagen sin tratar, tal y como es capturada por la webcam para poder tener los tonos del semáforo, y una vez comprobado esto da comienzo el tratamiento de la imagen.

El tratamiento de la imagen consiste en aplicar un filtro. En el apartado anterior, se daba a conocer que las imágenes carecen de complejidad al ser realizadas en un entorno virtual muy bien definido por lo que no es necesario la realización de complejos tratamientos, si bien, el filtro que se aplica es un filtro el cual está compuesto de varias tareas simultaneas.

El filtro aplicado es el llamado filtro Canny y gracias a la multitud de operaciones y a que la librería escogida para el tratamiento de las imágenes dispone de este filtro, se logra conseguir, con tan solo una operación, un resultado totalmente válido para la obtención de los datos necesarios. Más adelante se detallará el funcionamiento de este filtro.

Una vez aplicado el filtro comienza el proceso de búsqueda de los datos que a posteriori servirán para tomar las decisiones. En esta búsqueda de datos principalmente se intentan obtener referencias válidas para guiar al vehículo por el carril, en este caso, las líneas que delimitan el carril. Buscando estás líneas y obteniendo su progresión se puede determinar donde se encuentra la carretera y por donde debe ir el vehículo.

Con las líneas ya obtenidas, se pasa a buscar datos complementarios que sirvan para guiar el vehículo con mayor exactitud y una vez obtenidos dichos datos se procede al guardado de los mismos en una estructura que servirá posteriormente para decidir qué hacer, ya en el módulo de toma de decisiones.

Por supuesto este proceso se repite para las imágenes obtenidas de ambas cámaras, aunque cambie la perspectiva el proceso continúa siendo el mismo. Solo se producen cambios que en esta fase de diseño no procede conocer.

- Salida: la salida de este módulo no es más que la información obtenida de las imágenes que se han tratado. La salida varía un poco dependiendo del punto en que se encuentre el proyecto. Esto se estudiará con detalle en un apartado posterior.

MÓDULO TOMA DE DECISIONES

- Entrada: la entrada de este modulo se debe dividir en dos partes ya que, según el desarrollo del proyecto, se cambió a medida que se avanzaba en el mismo. El cambio de entrada radica en la arquitectura robótica seguida. En un principio el proyecto seguía una arquitectura reactiva, con lo que la entrada era simplemente los datos de una imagen que la webcam enviaba.

  Al cambiar el enfoque reactivo por el enfoque híbrido se incluyeron elementos propios de la arquitectura deliberativa y la entrada de este módulo pasó a ser un conjunto de datos que representaban los datos obtenidos de analizar varias imágenes suministradas por las webcams con los cuales deliberar qué acción debe realizar el vehículo.

- Proceso: al igual que en la entrada, en el proceso que se sigue en este modulo hay que diferenciar entre los enfoques reactivo e híbrido. En cuanto al enfoque reactivo se refiere, este módulo trabaja de manera sencilla, analizando los datos dados por el módulo de tratamiento de imágenes. En base a dichos datos se decide qué hacer e inmediatamente se lanza la ejecución de dicha acción.

  En ambos enfoques, la toma de decisiones se realiza mediantes un sistema basado en reglas. Según los datos que se tiene actualmente se comprueba qué decisión es la que más se ajusta a la situación y se ejecuta dicha decisión. Aunque posiblemente un sistema basado en reglas no sea el mejor sistema para esta situación, es un sistema que cumple con las expectativas y que funciona de una manera aceptable. Además en caso de necesitar más reglas para distintas situaciones, solo es necesario agregar la nueva regla y el resto de sistema seguirá funcionando de manera intacta.

  Cuando se incluyó el enfoque híbrido en el proyecto el proceso de toma de decisiones fue el que más se vio afectado por este gran cambio de filosofía. El módulo de toma de decisiones pasó a recibir una serie de datos en forma de muestras organizadas con las que tomar las decisiones. Para ello se estudia cada una de las muestras que se le han enviado por separado, dentro del límite que se establece, y las analiza. Una vez reconocido sus datos, comienza el proceso de decisión que dependerá de múltiples factores. Uno de estos factores es la máquina de estados, ya que el vehículo entrará en distintos estados dependiendo de la situación y lo que los datos indiquen. Los posibles estados se estudiaran más adelante con la ayuda de un gráfico. Teniendo en cuenta la situación actual del vehículo y la situación futura que muestran los datos obtenidos, se toma la decisión que el sistema entiende como más correcta. Se reitera que estas decisiones pueden ser erróneas pero tienen una mayor fiabilidad que las tomadas en el enfoque reactivo.

  Una vez que se ha tomado la decisión de qué hacer teniendo en cuenta todos los factores, se procede a ejecutar la orden, que simplemente es un mensaje al motor del vehículo indicando la potencia que debe aplicar sobre las ruedas.

- Salida: la salida es la misma ya sea en la etapa del enfoque reactivo o en la etapa del enfoque híbrido. En ambas etapas la salida es una orden al motor del vehículo indicando la potencia que debe ejercer sobre cada una de las ruedas.

4.2.2.- ARQUITECTURAS ROBÓTICAS

Como se ha comentado en anteriores puntos del documento, todo programa o sistema que tenga que ver con la robótica o los robots debe asentarse sobre las bases de una arquitectura o paradigma.

ARQUITECTURA REACTIVA

En el caso de este proyecto, no se eligió una arquitectura inicial y se avanzó sobre esa base, sino que comenzó el desarrollo del mismo sin una posición clara. Naturalmente los primeros prototipos generados tan solo eran capaces de realizar tareas simples aunque no por ello dejaba de ser un prototipo eficaz, al menos para las tareas a las que se le había programado.

Al principio, los prototipos desarrollados seguían los patrones definidos por las arquitecturas reactivas, es decir, el robot reaccionaba a los cambios en el entorno y a su posición. Con base a esta arquitectura los problemas según se intentaba ampliar el radio de acción del prototipo se multiplicaban. Cada vez era más impreciso ya que, en cuanto los datos obtenidos dejaran de seguir el patrón general, el vehículo reaccionaría a esos datos y tomaría la decisión de realizar una acción acorde con los datos analizados.

Si bien estos datos son correctos y están bien obtenidos, nos muestran un comportamiento erróneo de la situación y del entorno. Con esta arquitectura, el sistema no se detiene a pensar si los datos son lógicos o carecen de sentido, simplemente se dedica a tomar decisiones con los datos obtenidos en ese mismo instante.

Por lo que, teniendo cuenta lo comentado, era cuestión de tiempo que el vehículo, en alguna situación, ya fuera extrema o de total normalidad, realizara acciones que no fueran en concordancia con la situación real del entorno. Por ello, realizando las pruebas se observaban comportamientos extraños, erróneos. El índice de éxito en las pruebas era bajo, ya que casi siempre se encontraba con algunos datos que variaban en mucho el patrón que en ese momento seguía el entorno. Por ejemplo, era usual observar al coche tomar una curva y en medio de dicha curva interpretar que debe seguir recto con lo que la salida de la calzada se convertía en algo común a la hora de realizar las pruebas.

Aún así, los avances obtenidos con esta arquitectura se pueden calificar de notables ya que se obtuvo gran cantidad de información para analizar varios apartados. Por un lado, se veía claramente necesario la implementación de un sistema menos reactivo.

ARQUITECTURA HÍBRIDA

Aunque la arquitectura reactiva no lograba obtener una gran eficacia en los resultados, mostraba que el camino no era el equivocado y que simplemente se deberían cotejar más datos o de manera más minuciosa antes de actuar.

Por ello, se incluyó una parte deliberativa en el proyecto, una parte, la cual, decidiera si los datos que se están leyendo en este momento son datos fiables y que siguen un patrón o por el contrario son datos inválidos de una situación que se ha dado de manera casual en ese preciso instante. Con la inclusión de esta parte deliberativa y transformando el sistema a un sistema híbrido el aumento de fiabilidad se hizo notable desde la primera ejecución. Tan solo había que perfeccionar la novedad que se había incluido y el sistema lograría comprender las situaciones a las que el vehículo se enfrenta.

Además de incluir datos nuevos para deliberar qué debemos hacer, también se incluyo una máquina de estados la cual nos indicaría en qué situación se encuentra el vehículo. El objetivo de esta máquina de estados es hacernos diferenciar entre las distintas situaciones posibles ya que puede que haya diferencia si nos encontramos en recta o si estamos girando.

Como se puede adivinar, el diseño final de la aplicación está realizado sobre una arquitectura híbrida que permite deliberar que se debe hacer en el proceso de toma de decisiones gracias a la máquina de estados

y a las muestras almacenadas obtenidas de las webcams disponibles completando un sistema basado en reglas.

Las ventajas aportadas por esta arquitectura, principalmente, son fiabilidad y seguridad. Ya no se cambiará bruscamente de dirección en caso de encontrar una sola muestra que indique tal acción ya que el patrón seguido por los datos anteriormente obtenidos indica que estamos en una recta. Este es tan solo un ejemplo de las ventajas que aporta esta arquitectura a las situaciones que nos podemos encontrar a la hora de dirigir un vehículo.

4.2.3.- PROCESAMIENTO DE LA IMAGEN

Para procesar las imágenes obtenidas de las webcam que se disponen en el vehículo virtual se realizar un proceso compuesto por varias acciones o filtros que se aplicaran sobre la imagen. Estos filtros se obtienen de la librería AForge.net.

En el proyecto se utiliza un algoritmo de detección de bordes para hallar las líneas que se deben seguir. Gracias a la librería antes mencionada es posible realizar este algoritmo de una sola pasada. Por ello, y para no olvidar que este algoritmo se realiza de manera continua con cada imagen, se va a proceder a explicar en qué consiste, para saber por qué serie de filtros y operaciones pasa la imagen.

Este algoritmo fue desarrollado en 1986 por John F. Canny y tiene por objetivo la detección de contornos en las imágenes. Este algoritmo es considerado como uno de los mejores en su ámbito, la detección de dichos contornos, y se basa en la primera derivada y en la utilización de mascaras de convolución. Se encarga de hallar los puntos de contorno que no son más que zonas de píxeles en las que existe un cambio brusco de nivel de gris.

Este algoritmo se basa en tres criterios:

- Detección: criterio que expresa el hecho de evitar la eliminación de bordes importantes y no suministrar bordes falsos.

- Localización: establece que la distancia entre la posición real y la localizada del borde se debe minimizar.

- Respuesta: tal respuesta debe integrar las respuestas múltiples a un único borde.

Uno de los métodos relacionados con la detección de bordes es el uso de la primera derivada, la que es usada por que toma el valor de cero en todas las regiones donde no varía la intensidad y tiene un valor constante en toda la transición de intensidad. Por tanto un cambio de intensidad se manifiesta como un cambio brusco en la primera derivada, característica que es usada para detectar un borde, y en la que se basa el algoritmo de Canny (Valverde Rebaza, 2007).

El algoritmo de detección de bordes de Canny consiste en tres grandes pasos.

OBTENCIÓN DEL GRADIENTE

El objetivo en este paso es la obtención de la magnitud y la dirección del vector gradiente de cada pixel. Para ello, lo primero que se realiza es la aplicación de un filtro gaussiano a la imagen original con el objetivo de suavizarla. Una vez suavizada y mediante el uso de un algoritmo se haya el gradiente de cada pixel.

En este proceso se generan dos imágenes que indican la magnitud y la orientación del gradiente.

SUPRESIÓN NO MÁXIMA

El objetivo de esta fase es el adelgazamiento del ancho de los bordes obtenidos en la fase de obtención del gradiente. Con las dos imágenes obtenidas en la fase anterior, se realiza un procedimiento como sigue a continuación:

- Se consideran cuatro direcciones identificadas por las orientaciones 0°, 45°, 90° y 135° con respecto al eje horizontal. Para cada pixel se encuentra la dirección que más se aproxime a la dirección del ángulo del gradiente.
- Se observa si el valor de la magnitud del gradiente es más pequeño que al menos uno de sus dos vecinos en la dirección del ángulo obtenida en la imagen del paso anterior. De ser así se asigna valor 0 a dicho pixel, en caso contrario se asigna el valor que tenga la magnitud del gradiente.

La salida de este proceso es una imagen con los bordes adelgazados.

HISTÉRESIS DE UMBRAL

La imagen obtenida en el paso anterior suele contener máximos locales creados por el ruido. Una solución para eliminar dicho ruido es la histéresis del umbral.

El proceso consiste en tomar la imagen obtenida del paso anterior, tomar la orientación de los puntos de borde de la imagen y tomar dos umbrales, el primero más pequeño que el segundo. Para cada punto de la imagen se debe localizar el siguiente punto de borde no explorado que sea mayor al segundo umbral. A partir de dicho punto seguir las cadenas de máximos locales conectados en ambas direcciones perpendiculares a la normal del borde siempre que sean mayo res al primer umbral. Así se marcan todos los puntos explorados y se almacena la lista de todos los puntos en el contorno conectado.

Es así como en este paso se logra eliminar las uniones en forma de “Y” de los segmentos que confluyan en un punto.

Frecuentemente, es común que un cuarto y último paso se realice en el algoritmo de Canny, este paso consiste en cerrar los contornos que pudiesen haber quedado abiertos por problemas de ruido.

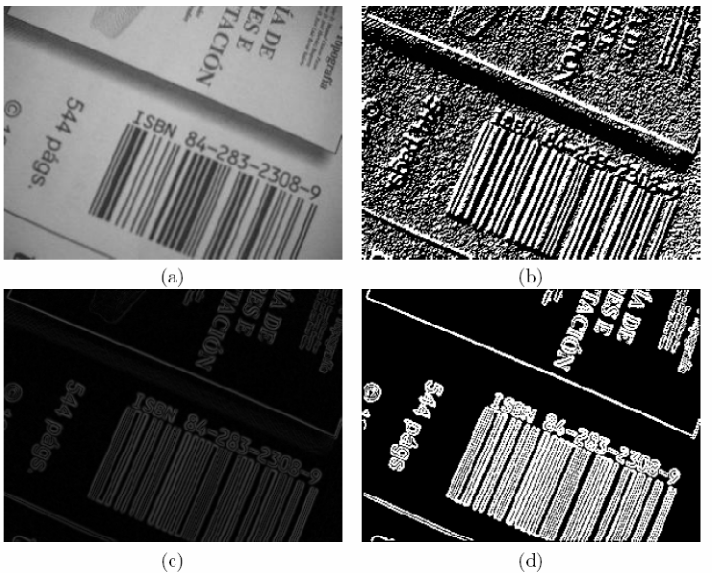


Figura 22: resultado de aplicar el detector de bordes de Canny: (a) imagen original; (b) orientación; (c) supresión no máxima; (d) histéresis de umbral

En la figura 22 se puede observar los estados por los que pasa la imagen hasta que finalmente se obtiene la imagen definitiva tratada con el filtro de Canny. Primeramente se realiza la obtención del gradiente (imagen b) en la que se observa el contenido de los bordes en relieve, posteriormente, y como indica el proceso más arriba, se procede a realizar la supresión no máxima (imagen c) en la que se obtienen los bordes aunque un poco oscurecidos y por último se realiza la histéresis de umbral (imagen d) para enfatizar los bordes y que sean más visibles.

4.2.4.- DISEÑO DETALLADO

En este apartado se estudiaran los diagramas que han ayudado a realizar el diseño. Por motivos de síntesis en el documento, solo se mostrarán los diagramas correspondientes al último de los prototipos, es decir, tal y como ha quedado el software una vez está finalizado. Los diagramas que ilustran el proyecto en la fase de diseño son los siguientes.

DIAGRAMA DE CLASES

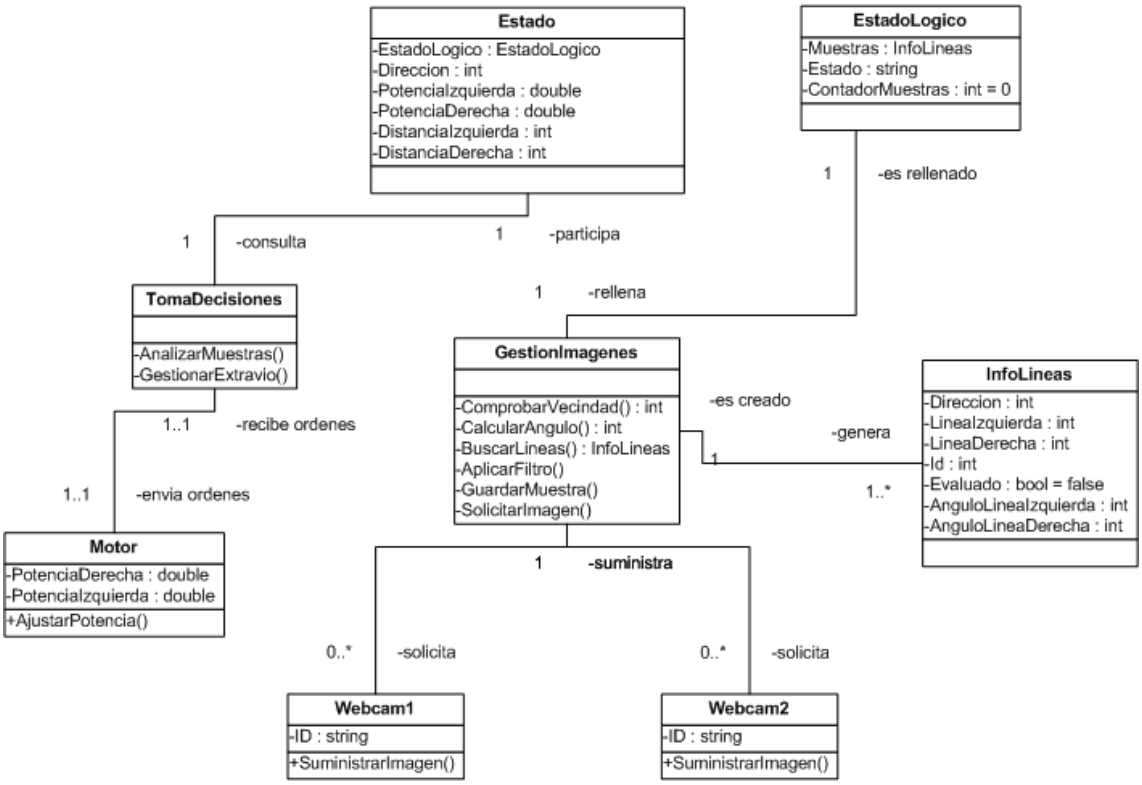


Figura 23: diagrama de clases

Con este diagrama de clases se quiere ilustrar la interacción entre las clases existentes en el sistema. Como se puede observar, se tiene un estado el cual contiene toda la información del vehículo en ese momento y un estado lógico que nos indica el comportamiento general del mismo. Los datos obtenidos por el módulo gestor de imágenes son guardados en el estado lógico y el módulo de toma de decisiones, ya con las muestras necesarias, tomará la decisión e enviará la orden al motor para que la ejecute.

En este diagrama se muestran dos webcam que contienen la misma información algo que no es óptimo pero en el paquete inicial proporcionado por RUC el sistema tiene dos webcams creadas independientemente, sin relación alguna entre ellas ni una clase padre de la que hereden información.

DIAGRAMA DE SECUENCIA

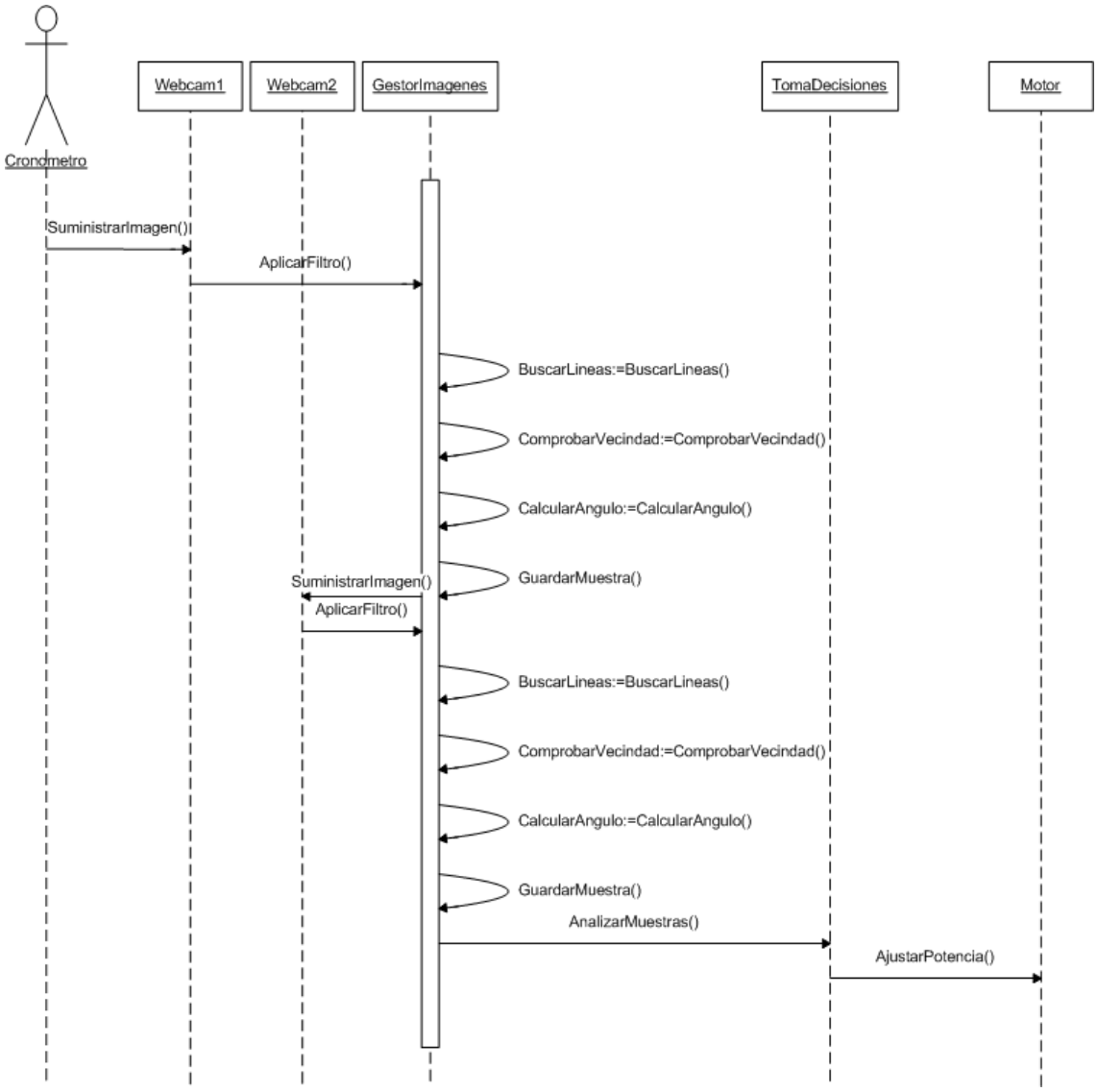


Figura 24: diagrama de secuencia

En este diagrama de secuencia, ilustra una iteración completa del sistema que se ha implementado en su versión final. El cronometro implementado (reloj en la figura) solicita cada cierto tiempo imágenes. Este cronometro solicita imágenes a la primera webcam y comienza el proceso de tratado de dicha imagen y, por último, el sistema gestor de imágenes se encarga de guardar la muestra en la estructura adecuada para posteriormente solicitar de ahí mismo la siguiente imagen a la segunda webcam, donde el proceso se repetirá hasta almacenar la muestra y pasar el testigo al sistema de toma de decisiones que analizará las muestras, siempre que sean suficientes, y tomará una decisión. Por último, el proceso termina enviado la decisión tomada al motor que se encargará de cambiar los valores necesarios para seguir dicha orden.

Este proceso se repite continuamente hasta que el sistema es detenido o se produce algún fallo.

MÁQUINA DE ESTADOS

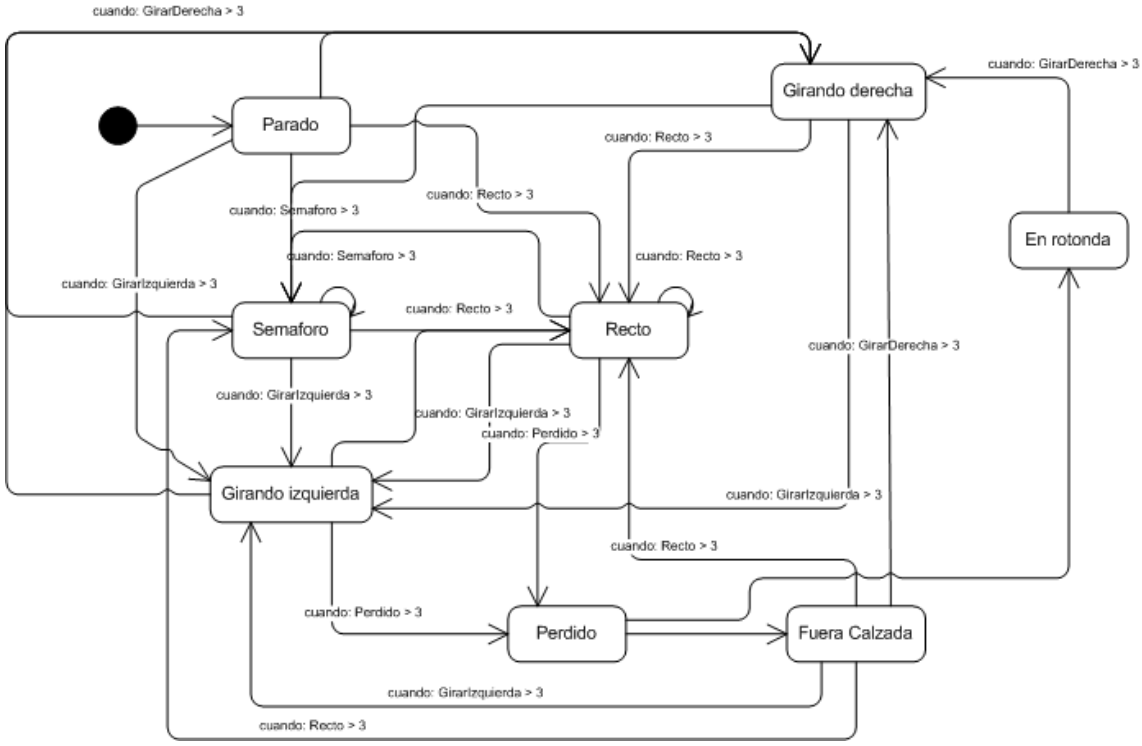


Figura 25: máquina de estados del sistema

Este diagrama correspondiente a la máquina de estados del sistema, ilustra todos los estados en los que el vehículo puede entrar y desde qué estado. En gran parte todos los estados son accesibles desde cualquier situación ya que el vehículo debe poseer la flexibilidad suficiente para afrontar cualquier situación o imprevisto.

Para el caso especial de las rotondas, primero debemos considerar el vehículo como perdido para posteriormente decidir que se ha entrado en una rotonda ya que es un caso especial.

4.3.- IMPLEMENTACIÓN

Al realizar la fase de implementación se ha ido dividiendo en prototipos con los que se observaban sus pruebas y se obtenían resultados que ayudarían a implementar el siguiente prototipo. Naturalmente cada uno de los prototipos no era implementado desde cero, sino que se coge como base el anterior prototipo y a partir de él, continuar el desarrollo hasta obtener los resultados deseados. Lo que se consigue con esta metodología es añadir pequeñas metas al desarrollo de manera que dicho trabajo se vuelva menos pesado y obligando a alcanzar dichas metas hasta llegar al producto final.

Como ya se comentó en el apartado de introducción relacionado con RUC, este proyecto parte con una estructura de código ya hecha, es decir, la comunicación entre el vehículo y los elementos tales como las webcams, el laser o el GPS ya está implementado en el *starter kit* de RUC. Esta implementación suministra un servicio totalmente funcional de manera manual, en otras palabras, con el *starter kit* proporcionado se podría mover el vehículo de manera manual y todos los elementos de dicho vehículo son funcionales pero no realizaría ninguna acción por sí mismo.

Esto ayuda a que todos los recursos se centren en lo realmente importante, que es la gestión de los datos obtenidos por los distintos módulos implementados y hacer un sistema automatizado sin necesidad de guiar al vehículo mediante ningún control.

Una vez introducida la metodología seguida se pasara a explicar la evolución del código organizado por prototipos. Por motivos de síntesis, se hablará de manera más notable de los prototipos principales o de los prototipos que produjeron cambios significativos en el transcurso del proyecto.

Antes de pasar en detalle al desarrollo del código conviene aclarar algunas convenciones que se han tomado para este proyecto:

- Todos los valores que tienen que ver con la izquierda se simbolizan con el número -1, es decir, que si el vehículo está girando hacia la izquierda, la dirección tendrá el valor -1. Por el contrario, los valores relacionados con la derecha se simboliza con el numero 1, mientras que la neutralidad se simboliza con el numero 0.

- Las imágenes que se obtienen de la cámara son de 128x128 pixeles y su origen de coordenadas se encuentra en la esquina superior izquierda mientras que el pixel 128,128 se encuentra en la esquina inferior derecha. Con lo cual, si descendemos en el eje de ordenadas estamos subiendo en la imagen y viceversa si aumentamos la coordenada “y”.

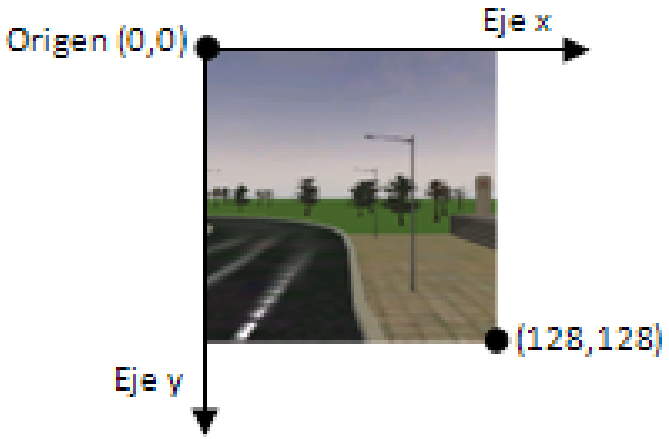


Figura 26: ejes de coordenadas en la imagen a tratar

4.3.1.- PROTOTIPOS REACTIVOS

Los prototipos que se describirán a continuación son los prototipos que se realizaron centrándose en el paradigma reactivo o arquitectura reactiva. Debido a la necesidad de sintetizar solo se mostraran los prototipos más importantes para el desarrollo del proyecto y los que mayor relevancia han tenido.

PRIMER PROTOTIPO

En esta primera iteración se dio comienzo al proyecto codificando las bases de cada uno de los módulos notables explicados en el apartado de diseño. Además, el proyecto comienza basado en una arquitectura reactiva con lo que los resultados se verán algo comprometidos.

MÓDULO GESTIÓN DE IMÁGENES

En este módulo se realizó la creación de las imágenes a tratar por la webcam. En este primer prototipo solo se hacía uso de una de las webcams y se gestionaba su información cuando dicha webcam se refrescaba, es decir, en esta parte del proyecto no se solicitaba información a la cámara, sino que ella misma, al refrescar el *frame* capturado suministraba la información.

Las imágenes obtenidas por la webcam eran guardadas para procesarlas de manera más cómoda. Una vez obtenida y guardada se procedía a la aplicación de los filtros necesarios. Ya en este punto el filtro aplicado era el correcto (véase Figura 27), el filtro de Canny, aunque aplicado de una manera estándar, algo que posteriormente cambiaría. Con este filtro conseguimos una imagen en blanco y negro donde el blanco simboliza los bordes de los objetos o elementos de la imagen y el negro la ausencia de ellos.

Además, ya desde el primer momento, se realizó el sistema que permitiría reconocer los semáforos. Sencillamente se encarga de analizar la imagen sin tratar y reconocer los pixeles con predominante color rojo. Dichos pixeles se contabilizan y se guardan para que posteriormente el sistema de toma de decisiones compruebe si se está ante un semáforo en rojo o se tiene vía libre.

Figura 27: imagen original y filtrada

Una vez obtenida la imagen, filtrada de manera correcta y comprobado que el semáforo se encentra en verde, o dicho de otra manera, no está en rojo, se pasa a la búsqueda de información dentro de la imagen. Este primer sistema de búsqueda de información intentaba localizar los pixeles blancos que estaban a una altura determinada de manera manual, o dicho de otra manera, se buscaban los pixeles por el eje “x” que se encontraban en una línea a una altura “y”, tomando los límites de la imagen como eje de coordenadas. Un problema que se obtenía con este método es que se conseguían pixeles que no aportaban nada de información, como por ejemplo, pixeles que estaban muy alejados del lugar donde el vehículo se encontraba. Ya en este prototipo, se mejoro ese aspecto delimitando los puntos obtenidos para conseguir una mayor eficacia.

MÓDULO TOMA DE DECISIONES

En este primer prototipo el sistema de toma de decisiones era muy arcaico y primitivo. Tanto es así que la única acción que realizaba era girar a la izquierda. Para ello lo que se comprobaba era la no existencia de puntos en el lado derecho del vehículo. Si en algún momento estos puntos desaparecían el sistema reaccionaría y ordenaría al vehículo girar a la izquierda.

Además de tan solo realizar una acción, dicha acción se realizaba con un grado de giro constante y que podría ser o no el idóneo para la curva que en ese momento se encontraba el vehículo.

Por último, y como se ha mencionado en el modulo de gestión de imágenes, este módulo ya es capaz de decidir si existe un semáforo que prohíba el paso o si se puede continuar la marcha. En caso de dictaminar que el vehículo se encuentra ante un semáforo en rojo se envía la orden de detener el motor, se anula la potencia ofrecida a las dos ruedas. En caso contrario, se continúa con la deliberación como si no existiera dicho semáforo.

CONCLUSIONES Y PROBLEMAS

Este primer prototipo tenía un enfoque de toma de contacto con todos los sistemas que interactúan, especialmente, el de comprobar las imágenes tomadas y su procesado. El sistema de toma de decisiones simplemente se integró de esta manera para ver que el coche respondía y cómo enviarle órdenes.

Partiendo de este prototipo como base se fue mejorando poco a poco con multitud de mejoras como la implementación de distintos sistemas de búsqueda de información en las imágenes, mejoras en el tratamiento de la imagen y sobre todo un sistema de decisiones más potente de manera que se adapte a todas las posibles situaciones. Tras un par de prototipos en busca del sistema de búsqueda de información perfecto, se llega al quinto prototipo que a la postre fue el último prototipo diseñado sobre una arquitectura única y exclusivamente reactiva.

QUINTO PROTOTIPO

Este prototipo significaría el gran giro que se producirá en este proyecto y que llevará, como se verá más adelante, a la solución final.

MÓDULO GESTIÓN DE IMÁGENES

Como se ha comentado previamente, el módulo de gestión de imágenes ha sido en el que más se ha invertido tiempo y esfuerzo. La obtención de las imágenes no ha cambiado, se sigue obteniendo la imagen de una sola cámara. Una de las diferencias es el cambio de parámetros en la aplicación del filtro, se sigue aplicando el mismo filtro pero con valores distintos lo que permite obtener más datos y más nítidos y fiables en la imagen tratada.

Una vez tratada se pasa a buscar información en ella que sirva para tomar una decisión. El sistema consigue crear una línea imaginaria siguiendo las líneas que delimitan el carril. Gracias al filtro de Canny sólo se tiene en la imagen tratada los bordes de los objetos y los elementos de distinto color, por lo que las líneas que delimitan el carril están perfectamente localizadas por este filtro.

Para hallar dichas líneas, se realiza un proceso dos veces, para cada una de las líneas que delimitan. Dicho proceso es el que sigue a continuación:

Paso 1: se parte de un pixel central estimado y se busca un pixel a su izquierda o a su derecha dependiendo de la línea a buscar. La coordenada “y” es un valor preestablecido que indica la altura a la que se debe empezar a buscar. Esta “y” origen, y como se verá más adelante, será la que guíe la búsqueda de los pixeles.

Paso 2: si se encuentra, el punto central deja de tomar importancia para centrarse en el pixel que se ha encontrado y se pasa al siguiente paso. En el caso de no encontrar dicho pixel, se repite el proceso dos veces más decrementando la coordenada “y” y si no se encuentra ningún pixel de esta manera, se considera que no existe línea en este momento actual.

Paso 3: a partir de la posición de dicho pixel se busca el siguiente en altura, es decir, decrementando la coordenada “y” y buscando en la misma coordenada “x” obtenida en el pixel anterior.

Paso 4: En el caso de no encontrar ningún pixel blanco en el punto (x, y-1), se busca en los pixeles colindantes incrementando y decrementando la coordenada “x” de igual manera para ir expandiendo la búsqueda de manera equilibrada por ambos lados de la coordenada “x” original. Es decir, se buscaría por orden en las coordenadas (x-1, y-1), (x+1, y-1) y así sucesivamente.

Paso 5: si aún así no existen pixeles blancos en esa coordenada “y”, una vez que el sistema se percata de que ha llegado al límite de la imagen, simplemente se busca decrementando nuevamente la coordenada “y”.

Se da por finalizado el proceso cuando se ha recorrido una altura de treinta pixeles en la coordenada "y", desde la coordenada "y" preestablecida al inicio del proceso. Por lo que los máximos valores que podemos obtener son treinta pixeles por cada línea.

Este proceso se repite dos veces, una vez para hallar la línea izquierda y otra para hallar la línea derecha. Los pixeles que se encuentran son guardados en dos estructuras, una para cada línea, de manera que se pueda acceder a ellos en el siguiente paso.

Una vez obtenidos dichos pixeles y guardados, se procede a buscar si dichos pixeles siguen un patrón de comportamiento formando una línea o simplemente son pixeles sueltos. Para ello se comprueba uno a uno los pixeles y siempre se comprueba con el anterior. Se realiza la comprobación observando si el pixel anterior es mayor o menor en cuanto a la coordenada "x" que el siguiente y este comportamiento se contabiliza en variables que ayudan posteriormente a decidir qué patrón sigue la línea. En caso de mostrarse una tendencia ascendente en los pixeles se considera la línea existe y se le asigna el valor 1 y en caso de mostrar una tendencia descendente se considera la línea con el valor -1, valor que simboliza descendencia en su avance.

Los valores de la tendencia que siguen las líneas es lo que se le envía al módulo de toma de decisiones que en este prototipo también ha tenido una serie de cambios que se detallaran en el siguiente apartado.

MÓDULO TOMA DE DECISIONES

Este módulo también ha tenido, desde el primer prototipo, una serie de cambios importantes como por ejemplo incluir giro para ambos lados, adaptarse a la curva y sus posibles cambios, etc.

Este módulo recibe los datos de una imagen, más concretamente los datos de las líneas que delimitan el vehículo en ese instante y con ambos datos y el estado actual del coche se delibera qué acción realizar. Una de las mejoras que se han implementado hasta este punto es la inclusión de un estado del vehículo en el cual se guarda la información de su dirección (girando izquierda (-1), girando derecha (1) y recto (0)), la potencia de cada rueda y la distancia desde el centro estipulado hasta cada una de las líneas.

Con la ayuda de estas variables y la información que el módulo anterior ha enviado se tomará una decisión y se ordenará su ejecución.

Para tomar la decisión, primero se comprueba el patrón que sigue la línea. Una vez comprobado dicho patrón se pasa a comprobar el estado actual del vehículo, ya que no es lo mismo si el vehículo está recto y los datos indican que siga recto o si el vehículo está girando y los datos indican que continuemos girando.

El caso más básico es el de continuar recto, ya que no se debe comprobar nada más, simplemente aplicar potencia. La única comprobación que se realiza aquí es la medición de la distancia del centro estipulado a las líneas. En el caso de que el vehículo se acerque mucho a la línea izquierda, en otras palabras, que entrase dentro del rango estimado, se realiza una corrección a la derecha, una pequeña ampliación de potencia en la rueda izquierda que hace que el coche se centre y vuelva hacia el centro del carril. De igual manera si se detecta una desviación a la derecha se aplica una corrección a la izquierda. Este caso solo se tiene en cuenta si el vehículo se encentra recto y deseamos seguir rectos. En otros casos la corrección no se tiene en cuenta.

En caso de querer realizar algún giro, el sistema se vuelve un poco más complejo ya que no sólo se aplica potencia a las ruedas sino que se intenta regular la potencia según la curvatura del giro. Para ello, una vez que se detecte que la imagen actual nos indica que giremos hacia algún lado, se comprueba si el vehículo ya estaba girando. En caso negativo, la solución es fácil, ya que se asigna una potencia estándar a aplicar a cada una de las ruedas que dirigen el vehículo y se envía la orden al motor. En caso de que el vehículo ya estuviera girando, aparte de seguir girando, se comprueba la distancia del punto central estimado hasta las líneas que haya. En este punto debemos diferenciar entre girar a la izquierda o a la derecha, ya que dependiendo de la dirección del giro una de las ruedas mantendrá la potencia que tenía cuando venía en línea

recta, es decir, solo se cambia el valor de una de las ruedas. Si estamos girando a la izquierda, en el caso de estar muy cerca de la línea derecha se incrementa la potencia que la rueda derecha tiene y en caso contrario si el vehículo está muy cerca de la línea izquierda. Si estamos girando a la derecha la rueda que sufre las modificaciones de su potencia es la izquierda y en el caso de estar muy cerca de la línea derecha se decrementa la potencia de la rueda izquierda y en caso de estar muy cerca de la línea izquierda se incrementa la potencia de dicha rueda.

En cualquier caso, este proceso siempre termina con la ejecución de una orden la cual transporta las potencias que se han decidido aplicar a las ruedas y que el motor debe aplicar. Una vez terminada la orden, todo este proceso vuelve a comenzar desde el módulo de gestión de imágenes.

CONCLUSIONES Y PROBLEMAS

Como se puede ver, se toma la decisión definitiva de qué se debe hacer con tan solo una imagen. Este hecho producía que en un determinado número de veces en plena curva el sistema con los datos proporcionados en la imagen interpretará que debe seguir recto o que al perder una de las líneas el sistema perdiera datos y se volviera poco fiable.

Estudiando los resultados y comprobando que las mejoras que se han implementado no son suficientes, se hace patente la necesidad de no tomar una decisión tan precipitadamente y que, aunque una imagen muestre que se debe cambiar de comportamiento, es recomendable comprobar que, efectivamente, de manera general, el comportamiento de la vía ha cambiado.

Aún así, el gran problema se encuentra en la toma de decisiones ya que la obtención de los datos es correcta. Aunque se mostrara un cambio de comportamiento en la curva cuando en realidad no lo hay esto es debido a que el sistema no es infalible y en determinados momentos puede obtener datos que, si bien, son correctos, no indican el estado general de la situación.

En otras palabras, por un dato o conjunto de datos puntual no se puede cambiar un estado general. Esto es lo que se llama enfoque reactivo.

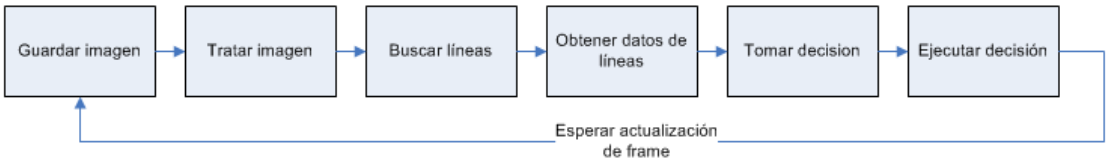


Figura 28: enfoque reactivo del sistema creado

Desde este punto, se empieza a plantear la necesidad de un sistema de toma de decisiones más eficaz que actúe no solo con un dato, sino con una serie de datos obtenidos en varios momentos distintos, aunque consecutivos, del mundo real. Es decir, obtener datos, almacenarlos y cuando se tenga una cantidad importante de datos de manera que muestren el patrón general que la vía sigue, deliberar qué acción se debe realizar y ejecutarla.

Por ello, quizás este sea el prototipo más importante que se ha realizado ya que muestra las carencias que una arquitectura puramente reactiva puede tener, demostrando en la práctica y sin ningún género de duda que las arquitecturas deliberativas y las híbridas pueden dar mejores resultados cuando el ámbito del problema es el seguimiento de un camino exacto. Puede que para ciertos problemas, la solución más rápida y eficaz sea un concepto reactivo pero en el problema que nos abarca queda totalmente demostrado que es necesario una arquitectura deliberativa que sea capaz de “razonar”.

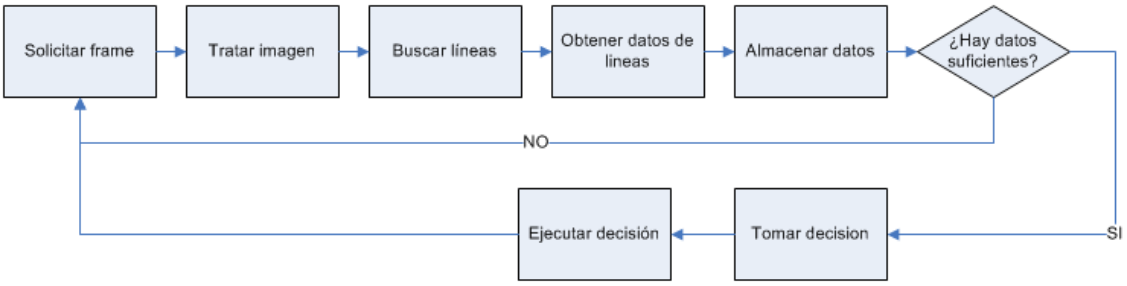


Figura 29: enfoque deliberativo aplicado al sistema

4.3.2.- PROTOTIPO HÍBRIDO

En este apartado, y a partir de aquí, se hablará de los prototipos que incluyen el paradigma híbrido como base. En definitiva, se comentará la inclusión de la parte deliberativa en el proyecto ya que la reactiva ya está implementada y se mostrará tan solo un prototipo ya que en esta fase sólo se crearon dos.

En este último prototipo se explica la inclusión de la arquitectura deliberativa en el proyecto y las mejoras que se han introducido a partir de él. Finalmente, la aplicación de dicha arquitectura se ha mostrado como la solución más eficaz ya que solo han existido dos prototipos deliberativos frente a los cinco prototipos reactivos. Se puede afirmar que el terreno avanzado con la utilización de la arquitectura reactiva no se ha realizado en vano ya que, fruto de ella, se ha logrado aprender mucho del comportamiento del vehículo y sus sistemas además de reutilizar elementos de esa arquitectura ya que, en base y salvo pequeños detalles que se comentaran, el módulo de gestión de imágenes no ha sufrido apenas cambios.

De forma general, se puede comentar que ya no se obtienen imágenes cuando la cámara se refresca, sino que se hace petición a la cámara del *frame* actual cuando se necesita. Para ello se ha creado un *timer* que se podría considerar como un elemento más del vehículo. Dicho *timer* lanza la petición de nuevo *frame* cuando termina de procesar la imagen actual más una espera de diez milisegundos. Con esto se logra gestionar más imágenes en menos tiempo ya que el sistema no espera a que la cámara refresque sino que se solicita el *frame* cuando se necesita y, simultáneamente, no se entorpece el funcionamiento del sistema ya que no se solicita otra imagen hasta diez milisegundos después de que haya terminado de gestionar la anterior.

ÚLTIMO PROTOTIPO

MÓDULO GESTIÓN DE IMÁGENES

Excepto pequeñas mejoras, el módulo de gestión de imágenes poco ha cambiado en cuanto al proceso de tratado de imagen y búsqueda de información.

El principal cambio introducido es la adaptación para almacenar los datos obtenidos en el proceso en lugar de enviarlos directamente al módulo de toma de decisiones. En este sentido, lo que se ha hecho es almacenar los datos temporalmente para, una vez terminado el proceso de búsqueda, guardar dichos datos junto con los que se obtuvieron con anterioridad, si los hubiera. Además de esto, ya no se espera a que la webcam actualice el *frame* sino que el mismo modulo solicita el *frame* deseado en el momento que lo necesita para gestionarlo. Se realizan dos peticiones, una por cada cámara y en cada ejecución de este módulo se obtienen datos de dos imágenes, una por cada una de las cámaras existentes en el vehículo. Con ello se consigue obtener desde otra perspectiva distintos datos lo que hace más fiable la decisión que posteriormente se tome en el siguiente modulo.

Como cada cámara posee una perspectiva distinta se ha hecho necesario adaptar el algoritmo de búsqueda de líneas, que no deja de ser el mismo, pero adaptado en cada caso dependiendo de la cámara que

se esté tratando. De esta manera, se sintetiza el número de líneas de código con unas pequeñas adaptaciones del sistema de búsqueda de líneas ya existente.

El último cambio destacable de esta parte del sistema es la obtención del ángulo que forman las líneas halladas con una línea recta que recorre el eje "y". El cálculo de dicho ángulo se realiza por triangulación con la ayuda del teorema de Pitágoras[5], el teorema del seno[6] y tomando las coordenadas de los pixeles como puntos del triangulo formado se calcula el ángulo que interesa y se almacena en una estructura que posee más datos de interés y que, a la postre, junto con el resto de estructuras almacenadas, será con lo que se tome la decisión final.

Como se puede observar imagen (véase Figura 30) se conocen dos puntos del triangulo, con lo que formar el triángulo rectángulo es directo. Teniendo las coordenadas de todos los puntos es sencillo calcular la longitud de los catetos y posteriormente la longitud de la hipotenusa con el teorema de Pitágoras. Una vez calculado esto, obtener el ángulo buscado es sencillo dando que se posee la longitud de la hipotenusa y el ángulo opuesto a la hipotenusa, que es noventa grados. Con estos datos se aplica el teorema del seno y se obtiene el valor del ángulo deseado.

Estos ángulos se interpretan inversamente, es decir, cuanto más pequeño es el ángulo, más grado de giro o de potencia en la rueda se ha de aplicar e, inversamente, cuanto menos ángulo haya menos potencia se ha de aplicar para girar. Para hacer que este cálculo sea inverso y se obtenga un valor final mayor con un valor menor del ángulo se ha utilizado el cálculo de la inversa del ángulo multiplicada por dos.

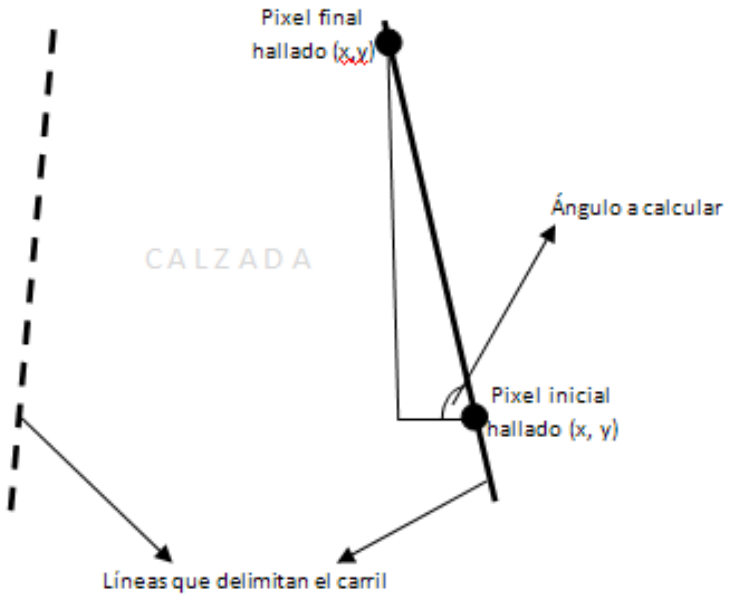


Figura 30: cálculo del ángulo formado por la línea

El cálculo de este ángulo es lo último que se realiza con la ayuda de la imagen, ahora, con todos los datos almacenados en la estructura correspondiente, se guardan para que el modulo de toma de decisiones consulte estas muestras y delibere.

[5] Teorema de Pitágoras: http://es.wikipedia.org/wiki/Teorema_de_Pit%C3%A1goras
[6] Teorema del seno: http://es.wikipedia.org/wiki/Teorema_del_seno

Antes de enviar a deliberar, este proceso se repite con la imagen que se obtiene de la otra cámara, se solicita dicho *frame* y este proceso íntegro se realiza también con la otra imagen. Una vez se termina de procesar las dos imágenes es cuando el módulo de toma de decisiones entra en acción.

MÓDULO TOMA DE DECISIONES

Quizás el módulo de toma de decisiones es el más afectado debido a la inclusión de la arquitectura híbrida. Anteriormente a este concepto, este módulo tan solo debía comprobar los datos de una imagen y en base a lo que veía actuar. Ahora, con la introducción del enfoque deliberativo este sistema se ve obligado a comprobar las muestras almacenadas por el modulo de gestión de imágenes y comprobar los datos que indican cada una de ellas. Con estos datos el sistema será capaz de obtener el patrón que la carretera sigue y tomar una decisión mucho más fiable y acertada.

Primeramente, se comprueba todas las muestras una a una. En el caso de proyecto el número de muestras a analizar simultáneamente se ha limitado a cinco. Diversas pruebas han dado como resultado este número indicando que es suficiente para decidir qué debe hacer el vehículo. Una vez analizadas estas muestras se determina que patrón siguen y en base a ello decidir.

Como de costumbre, se decidirá con el patrón que las muestras han indicado y con la información que se posee de la situación actual del vehículo. Esta última información ha cambiado un poco, de nuevo intentando adaptarse al enfoque deliberativo introducido. Estos cambios tienen que ver en la introducción de un estado lógico como si el vehículo fuera una máquina de estados. Anteriormente se poseía un estado pero este cambiaba de manera reactiva con cada decisión del vehículo, este nuevo estado lógico solo cambiará cuando, efectivamente, el coche cambie su comportamiento.

Por lo tanto, teniendo en cuenta ambos datos, tanto las muestras como el estado lógico, se tomará una decisión. En este caso, al igual que en el enfoque reactivo la decisión más cómoda es la de seguir recto, que una vez deliberado que se desea ir recto, simplemente se aplicará la potencia deseada y mandará la orden, aparte de cambiar el estado lógico si es que este no se encontraba en “Recto”.

A la hora de realizar giros se encuentra otro de los grandes cambios que se han producido en el proyecto y que ha hecho que la fiabilidad del giro aumente de manera sustancial y es el cálculo de potencia a aplicar a la rueda que hará que el vehículo gire teniendo en cuenta el ángulo formado por las líneas, ángulo calculado previamente en el módulo de gestión de imágenes.

Ahora, cuando las muestras detallan que se debe realizar un giro se comprueba el ángulo almacenado y mediante un cálculo fijo se ajusta el giro. Dicho cálculo obtiene el valor que se le ha de añadir a la potencia base, que es 0.3, para girar. Dependiendo del giro, este cálculo se aplica a la rueda izquierda o a la rueda derecha. Este cálculo se realiza siempre que las muestras indiquen que se debe girar, se calcula la potencia con los ángulos guardados en las muestras para actualizar el giro y comprobar que el vehículo va por el buen camino.

En el caso de indicar las muestras que se debe seguir recto, si el vehículo se encontraba en algún estado de giro se enderezará poco a poco quitando potencia sobre la rueda que posee más. De esta manera el vehículo se endereza de una manera más natural además de adaptarse mejor a su nuevo estado y no de una manera brusca.

Por último se ha gestionado los posibles extravíos teniendo en cuenta que el vehículo puede salirse, al igual que, en un caso determinado, una persona real podría hacerlo. Para ello se ha contemplado esta posibilidad añadiendo como posibles estados dentro de la máquina de estados. Cuando las muestras dan el coche por perdido, o dicho de otra manera, la mayoría de muestras del almacén no tienen datos de las líneas, se busca qué líneas fueron las últimas de las que se tuvieron constancia. En el caso de que la última línea fuera la izquierda quiere decir que se ha salido de la calzada por la derecha por lo tanto se aplica un giro fijo y brusco a la izquierda para poder volver a la vía. Por el contrario si la última línea de la que se tuvo información es la derecha se aplica el mismo giro que el anterior pero hacia la derecha para intentar recuperar dicha línea.

Existe un caso especial en el que las líneas que han sido halladas por última vez son las dos que delimitan el carril. En estos casos se considera que el vehículo se encuentra ante algún tipo de intersección o rotonda con lo que se aplica un giro a la derecha constante para lograr pasar dicha rotonda.

Esta solución creada para las rotondas se tomó debido a la amplitud del proyecto. Para relajar el trabajo e intentar delimitar el proyecto, se tomó esta decisión en el desarrollo del sistema. En un futuro, gestionar las rotondas con un método avanzado sería una posible mejora del proyecto.

5.- RESULTADOS

En este apartado se comentaran las pruebas realizadas con los distintos prototipos y su evolución a medida que los módulos se mejoraban. Estas pruebas serán las que demuestren si el proyecto funciona, es válido y cumple con los objetivos marcados o si, por el contrario, no se ha logrado obtener una implementación que de solución al problema.

Se van a pasar por alto las pruebas realizadas en el primer prototipo ya que, aunque es un prototipo muy representativo, las pruebas de dicho prototipo son irrelevantes por su escasa eficacia y no se pueden comparar al resto del proyecto.

Como se ha comentado con anterioridad, el primer prototipo solo era capaz de decidir si se debía girar a la izquierda única y exclusivamente y estaba hecho para tomar un primer contacto con el sistema y con todos los elementos que en un futuro se utilizarían. A modo de curiosidad, cabe destacar que, con ese primer sistema, el cien por cien de las veces lograba girar a la izquierda pero con un grado de giro fijo, y sin calcular si giraba demasiado y muchas otras comprobaciones que en prototipos más avanzados si se realizan. A partir de este primer prototipo se fueron implementando mejoras, siempre dentro del marco reactivo, empeorando el porcentaje de efectividad de giro pero aumentando la fiabilidad del conjunto.

5.1.- PRESENTACIÓN DE LAS PRUEBAS

Las pruebas que serán presentadas a continuación vendrán acompañadas siempre de imágenes, tanto de la posición del vehículo desde una vista de seguimiento como las imágenes que las cámaras capturan. Estas imágenes serán de gran ayuda para ilustrar cada uno de los casos. Las imágenes mostradas que han sido capturadas por las cámaras web no solo se limitarán a ser las imágenes reales, sino que se mostrará también las imágenes que realmente se han procesado en el módulo de gestión de imágenes, es decir, las imágenes ya filtradas.

Además de las imágenes filtradas, en determinados casos se mostrarán las imágenes con las líneas que el módulo de gestión de imágenes ha obtenido, de manera que sea más claro entender el comportamiento del vehículo. Estas líneas halladas se interpretaran como pixeles pintados de rojo y de verde en la imagen. Los pixeles rojos representan la línea izquierda y los pixeles verdes representan la línea derecha. Además se pueden observar algunos pixeles azules los cuales representan el centro precalculado que el algoritmo tiene en ese momento, ya que este centro es cambiante dependiendo de la situación del vehículo.

Dada la importancia del hardware gráfico y del número de *frames* por segundo de los que se dispone, se mostrará, gracias a un software externo e independiente al proyecto, el número de *frames* por segundo en el momento de la prueba, recordando que treinta *frames* por segundo es el límite en el que el ojo humano aprecia discontinuidad en las imágenes, es decir, por debajo de treinta *frames* se comienza a observar ralentización en el movimiento, y, se podría decir, que es el número optimo de imágenes por segundo en el que el proyecto debería trabajar.

A parte de los *frames*, también se muestran otra serie de datos que llegan a ser ilegibles y que no son demasiado relevantes para el transcurso del proyecto como son la temperatura del núcleo del chip gráfico y la temperatura de las memorias gráficas. Estos datos no son relevantes para el proyecto ni para las pruebas, si bien el software externo utilizado no permite que se oculten dichos datos.

Respecto a las estructura de las pruebas, se diferenciaran entre prototipo reactivo y el híbrido y además se comprobará uno a uno los ámbitos más importantes de la carretera, es decir, circulando de manera rectilínea, giros, semáforos y rotondas, aunque este último caso solo en el prototipo híbrido.

Finalmente se estudiará de manera comparativa los resultados de ambos enfoques y se demostrará con números objetivos que el sistema híbrido es el correcto para encontrar la solución al problema que se aborda.

5.2.- PRUEBAS PROTOTIPO REACTIVO

Llegados a este punto, el prototipo tiene la capacidad de decisión suficiente para abordar una carretera que guarda continuidad, es decir, este prototipo no gestiona cruces ni rotondas, pero es capaz de seguir una carretera a través de sus líneas. Dentro de este marco las pruebas se limitan a un corto espacio, una línea recta y un giro a la izquierda hasta llegar a la primera rotonda en la cual el vehículo podría reaccionar de la manera más imprevista.

Como se ha comentado en el apartado de implementación, en este prototipo reactivo solo se ha utilizado una de las dos webcams que están disponibles, con lo que las imágenes de prueba solo serán suministradas por dicha cámara.

5.2.1.- CIRCULANDO EN LÍNEA RECTA

Esta fase de pruebas tiene como objetivo demostrar la eficacia o no del prototipo cuando estamos circulando en línea recta.

CASO 1

El primer caso presentado en esta parte de las pruebas será el del vehículo en línea recta circulando con normalidad. En él el vehículo se encuentra en un estado idóneo, ya recto y con un tramo recto sin complicaciones añadidas.

Figura 31: caso 1: circulación en línea recta

Como se puede observar en la imagen (véase Figura 31) el vehículo se encuentra en una situación normal dentro de una línea recta y circulando de correctamente entre las líneas que delimitan el carril. Dichas líneas se pueden observar en la imagen pequeña de la esquina inferior derecha donde se muestra una línea roja y una línea verde y se verifica que el sistema está encontrando unas líneas que le indican al vehículo que debe continuar recto. Además se puede observar el punto central a partir del cual se han obtenido las líneas.

Esta situación representa lo que sería deseable en el caso de encontrarnos en tramos rectilíneos y es el objetivo de este proyecto. Además de estar perfectamente colocado el vehículo, como vemos en la parte superior, se dispone de 29 *frames* por segundo, es decir, prácticamente un número óptimo de *frames*.

CASO 2

En este caso se va a presentar la situación del vehículo pasando por un paso de cebra y la influencia de la confusión de las líneas seleccionadas en casos como este, en los que, para el sistema, es simplemente un montón de pixeles colocados en el centro y que todos pueden formar la línea buscada.

Figura 32: caso 2: paso de cebra

Como se puede observar en la imagen (véase Figura 32), el vehículo ya se encuentra desplazado hacia la derecha debido a que detecta las líneas arbitrariamente dependiendo de la situación exacta del vehículo. En dependencia de qué pixeles obtenga el vehículo puede reaccionar de una manera u otra. En este caso concreto se puede observar que en el lado derecho hay un carril extra pero este carril no lleva a ninguna parte puesto que en unos metros este mismo carril termina.

El problema radica en que, aunque se disponga de un punto central, los pixeles seleccionados podrían estar indistintamente a la derecha o a la izquierda de dicho punto central, algo que en futuros prototipos se mejora. Además, el prototipo reactivo cuando encuentra la más mínima variación, la ejecuta sin pensarlo con lo que, en este tipo de situaciones, se vuelve muy inestable. De ser un prototipo deliberativo, esperaría a que hubiera suficientes muestras para deliberar que, efectivamente, la situación de la carretera ha cambiado.

Otra de las variables que podría influir en la situación son los *frames*. En este caso se disponen de unos cómodos 27 *frames* que no son óptimos pero se acercan mucho a la medida óptima con lo que no presentan ningún problema ni afectan al resultado final de la prueba.

CASO 3

En este tercer caso se probaran las correcciones que se han implementado para que el vehículo las realice única y exclusivamente en una línea recta. Estas correcciones las realizará cuando el punto central entre dentro de un rango estipulado y ordenará ejecutar un pequeño cambio de dirección que hará que el vehículo se centre.

En la prueba mostrada, se ve como el vehículo ha realizado una corrección por que ha detectado que se encontraba muy cerca de la línea izquierda. Tras esta corrección el vehículo se ha descentrado se está desplazando ligeramente hacia la línea derecha. En el momento que detecte que se encuentra muy cerca de esa línea corregirá hacia la izquierda.

Figura 33: caso 3: corrección en línea recta

Las correcciones realizadas son muy pequeñas y delicadas, son prácticamente inapreciables y aunque el coche no va dando bandazos de lado a lado del carril si hacen que vaya realizando diagonales no muy exageradas. El uso de esta corrección es sobre todo en la salida de las curvas donde puede que el vehículo no haya quedado centrado sobre el carril correspondiente. Con estas correcciones se pretende centrar el vehículo para que se encuentre en las mejores condiciones para atacar el siguiente giro con mayor comodidad.

En la imagen (véase Figura 33), se puede observar como el vehículo está más cerca de la línea derecha y llegado el momento realizará una corrección a la izquierda para no invadir el carril contrario. Más claro se puede observar en la imagen que nos muestra el punto central ya que se ve como está más cerca de la línea derecha que de la izquierda.

Por último, se puede observar cómo en esta prueba se está en una situación más precaria en cuanto a *frames* se refiere. En el momento de la realización de la prueba, se disponían de 18 *frames* lo que podría provocar una lentitud en las reacciones y hacer que el vehículo tarde un poco más en reaccionar. A pesar de ello, para la prueba que hemos realizado tener una lentitud en este aspecto no es muy determinante ya que como se ha comentado, el vehículo avanza hacia la línea derecha de manera

constante y sin brusquedad con lo que no existe la posibilidad de que se salte la corrección correspondiente a pesar del bajo rendimiento gráfico.

CONCLUSIONES

Analizadas las posibilidades más comunes en este prototipo reactivo y en lo que a circulación en línea recta se refiere, se puede decir que el vehículo se defiende bastante bien recordando que estamos ante un paradigma reactivo, es decir, la mínima variación haría que el vehículo cambiase de dirección. Precisamente este hecho es el que hace descender la efectividad del conjunto ya que, aproximadamente, en el 50% de los casos, cuando se encuentra ante un paso de cebra se vuelve inestable e intenta girar a la derecha. En el resto de situaciones el conjunto responde bien, incluso en la salida de las curvas, gracias a las correcciones el vehículo no circula en mitad de dos carriles sino que intenta ceñirse a un único carril.

Teniendo en cuenta estas conclusiones se puede decir que la efectividad de este prototipo a la hora de atacar las rectas es de un 65% ya que, aunque en las situaciones de paso de cebra sean una pequeña lotería, es cierto que la cantidad de ellos comparada con la circulación normal es menor con lo que conforman un porcentaje menor al resto de circulación general.

5.2.2.- SEMÁFORO

Aunque en el semáforo el comportamiento parezca trivial, hay situaciones que pueden variar el éxito o fracaso del proyecto.

CASO 1

En este primer caso se comprobará la eficacia del sistema creado para que el vehículo quede detenido ante el semáforo. Se propone una situación en la que el semáforo está en rojo con suficiente antelación como para que el vehículo detecte que se encuentra ante un semáforo rojo y se detenga a tiempo con seguridad.

En esta situación el sistema ordena parar el vehículo y continúa comprobando las imágenes que le van llegando para ver si el semáforo cambia de color o continúa en rojo. Mientras el semáforo sigue en rojo el vehículo no se mueve, está detenido y en el momento en que el semáforo deja de ofrecer pixeles rojos, el vehículo comienza andar, que es la orden que el sistema ha querido ejecutar tras el semáforo en rojo.

Como se muestra en la imagen (véase Figura 34), se ve el vehículo parado antes el semáforo, la imagen original capturada por la webcam donde se muestra el semáforo en rojo con sus dos focos, el que se encuentra a media altura y el que se encuentra en lo alto y, por último, se puede observar la imagen tratada con los puntos rojos impresos en dicha imagen.

Por otro lado, como se puede comprobar en la esquina superior derecha de la imagen grande, se observa que la prueba se ha hecho con 18 *frames* por segundo que no es un número idóneo de imágenes por segundo pero para el enfoque reactivo es más que suficiente ya que con encontrar una sola imagen donde se observe el semáforo en rojo, el módulo de toma de decisiones ya ordenará al vehículo detenerse con lo que para situaciones como estas, la exigencia de *frames* no es tan grande como en otras situaciones un poco más delicadas.

Figura 34: caso 1: vehículo parado ante el semáforo

CASO 2

En este caso se representa una escena en la cual el semáforo se ha empezando a cambiar justo cuando el vehículo estaba pasando por debajo de él, prácticamente. En esta situación es previsible que el enfoque reactivo trabaje mejor, ya que en el momento que detecta rojos, se envía la orden de detección, pero hay casos, al igual que en el mundo real, en el que es materialmente imposible detenerse a tiempo sin saltarse el semáforo.

Como demuestra la imagen (véase Figura 35), mientras el vehículo estaba circulando el semáforo comenzó su proceso para pasar de verde a rojo. Ya en el ámbar o amarillo el sistema reacciona frenando el vehículo pero en este caso como se puede ver ha sido imposible incluso con un paradigma reactivo, que instantáneamente sin necesidad de más pruebas detendría el vehículo.

En este caso el sistema comprobó la imagen y detectó una cierta cantidad de pixeles rojos, que están mostrados en la imagen tratada. Con esos pixeles rojos, se tomó la decisión de detener al vehículo, aunque como se aprecia en la imagen grande, el vehículo está en una situación incorrecta, en mitad de paso de cebra entorpeciendo el posible cruce de viandantes por dicho paso.

Para esta prueba se ha dispuesto de 28 *frames* por segundo con lo que no ha habido problemas respecto a dicha variable y el paradigma reactivo ha reaccionado de una manera muy rápida, sin poder evitar que el vehículo invadiera el paso de cebra.

Figura 35: caso 2: semáforo saltado

Para estos casos el error es tolerable, aunque no deja de ser un error. Pero bien es cierto, que en determinados cosas en el mundo real esto también sucede y no por negligencia del conductor sino por condiciones de la vía y vehículo por ello se asume que es un error pero se considera inevitable.

CONCLUSIÓN

En el ámbito de los semáforos se considera que el sistema reacciona correctamente y que detiene el vehículo constantemente y de manera segura hasta que el semáforo vuelve a ponerse en verde. Es cierto que no se muestra un cien por cien de efectividad pero es prácticamente imposible conseguir ese cien por cien por la incapacidad de adivinar en qué momento cambiará el semáforo de estado. De ser conocida esta variable se estaría ante un error más grave pero como se desconoce este dato se considera este error de baja gravedad con lo que se puede afirmar que el sistema funciona en estas situaciones a un noventa por ciento.

5.2.3.- GIRANDO

Este punto constará de las pruebas realizadas en el entorno virtual cuando el vehículo se encuentra girando en curva. Debido a lo limitado del mapa las pruebas se realizaran sobre una misma curva. Es reseñable que todas las curvas dispuestas en el mapa de RUC son exactamente iguales algo que no se ha aprovechado, sino que se ha intentando hacer lo más adaptable posible al vehículo de manera que en otras situaciones con otras curvas reaccione de manera correcta.

CASO 1

En este primer caso se representa la situación de un giro normal corriente, situación en la que el vehículo reacciona al leer las líneas obtenidas e intenta girar con más o menos acierto según los casos.

Figura 36: caso 1: giro normal con el vehículo respondiendo correctamente

En esta imagen se puede ver como el vehículo interpreta la curva mediante las líneas que se obtienen en el proceso de gestión de imágenes y determina que se encuentra ante una curva a la izquierda. Para ello manda la orden de aplicar potencia sobre la rueda derecha y comienza a girar. En este prototipo la solución para estos giros era con el cálculo de distancias entre el punto central y las líneas una vez ya nos encontramos en la curva. Con este cálculo se aplicaba más o menos potencia algo que en futuros prototipos dejó de existir. Como se puede ver en la imagen grande el vehículo invade mínimamente el carril contiguo aunque se debe señalar que los carriles diseñados en la ciudad virtual son realmente estrechos y solo cabe justo el vehículo.

Si bien, este es el estado deseable para toda la curva, desgraciadamente, para este prototipo no es posible asegurar que el vehículo consiga realizar la curva debido a que casi continuamente está cambiando su grado de giro para adaptarlo a la curva. En el momento que se obtenga una imagen con datos anómalos a los que están sucediendo se tomará una decisión con la que probablemente se terminaría fuera del asfalto de manera irremediable.

Obviamente hay un porcentaje el cual es aceptable ya que incluso un ser humano tampoco garantiza el cien por cien de las curvas realizadas correctamente por lo que existe un margen permitido aunque este prototipo no llegue a dicho margen.

En estos casos el rendimiento del sistema hardware donde se ubique el prototipo es de suma importancia ya que aquí es cuando se necesita obtener los datos lo más rápidamente posible para poder reaccionar y rectificar o ajustar el giro. Si se dispone de una buena respuesta hardware, el rendimiento del paradigma reactivo en estos casos aumenta aunque no es perfecto, como se comprobará en otros ejemplos.

CASO 2

En este caso se representa una salida de la calzada en la curva aún con buena respuesta hardware debido a la excesiva rapidez en la toma de decisiones del paradigma reactivo.

Figura 37: caso 2: vehículo a punto de salirse de la calzada

En la imagen se observa al vehículo subido al bordillo y apunto de seguir recto hacia el exterior de la curva que se estaba trazando. En este caso, la mala colocación del vehículo es debido a la rapidez con la que el paradigma reactivo ha tomado la decisión de girar.

Este prototipo ha reaccionado al reconocimiento de la curva en cuanto se detectó la misma lo que ha hecho que el vehículo girara demasiado pronto. Intentando corregir ese giro prematuro se ha intentado rectificar con lo que el vehículo se ha puesto recto dentro de la curva y al comenzar a leer de nuevo si había curva o no era demasiado tarde. Como se puede apreciar en la imagen, la cámara web del vehículo perdió una de las líneas de referencia, incluso toma la supuesta línea que delimita el carril por la derecha como línea izquierda con lo que, en este punto, es imposible corregir la situación y el vehículo finalmente acabaría fuera de la calzada.

Incluso, en este caso, ni la buena respuesta del hardware en el momento de la salida de la calzada ha podido evitar dicha salida. No se ha logrado reaccionar a tiempo a una situación tan adversa provocada, como se ha comentado, por la prematura decisión de tomar la curva.

CASO 3

Este último caso es difícil que suceda en este prototipo ya que la carencia del mismo es que se sale mucho de la calzada por el exterior de la misma. Sin embargo, no queda exento de otra serie de anomalías a la hora de tomar las curvas como es la invasión del carril contiguo o incluso del carril contrario.

Como digo, este prototipo presenta casos, aunque muy limitados, de invasión del carril contiguo y es muy raro ver el vehículo invadiendo el carril contrario, con lo que, en este sentido, los resultados de este prototipo son bastante aceptables.

Figura 38: caso 3: invasión carril contiguo

La invasión del carril contrario es debido a que, una vez que se ha girado en exceso, el punto central que sirve de guía para hallar las líneas que delimitan el carril se posiciona en el carril contiguo con lo que las líneas con las que se empieza a guiar desde ese momento son las líneas del carril contiguo y a partir de él continuar con la marcha. Si por alguna razón el vehículo de nuevo girase en demasía, invadiría el sentido contrario y tomaría las líneas que delimitan el carril como las líneas del sentido contrario, es decir, una vez que el vehículo cambia de carril es complicado que él solo vuelva al carril original, solo un cambio del estado general de la carretera (volver a una recta, una nueva curva) podrían ocasionar la vuelta al carril original del vehículo.

También para este caso se ha dispuesto de un número óptimo de *frames* por lo que el hardware ha respondido a la perfección a esta fase de pruebas lo que ha permitido que las pruebas sean totalmente validas y nada influenciadas por el rendimiento del hardware.

CONCLUSIONES

Las conclusiones que se pueden sacar de este apartado de pruebas es que, de manera general, el vehículo reacciona con demasiada antelación a las curvas, es decir, que el paradigma reactivo es demasiado precipitado en la toma de decisiones por lo que se ve necesario la introducción del paradigma deliberativo en el proyecto, algo que asegure que se encuentra ante una curva.

Además, se puede concluir que este prototipo tiende a salirse de la curva por el exterior, sin embargo, es más improbable invadir el carril contiguo. En un caso extremo, es preferible que el vehículo invada el carril contiguo a que se salga de la calzada, ya que si el vehículo se sale de la calzada significa el fin del trayecto, cosa que no ocurre si se invade el carril contiguo, ya que el vehículo sigue circulando mediante un sistema guiado por las líneas.

Teniendo en cuenta estos datos la fiabilidad del conjunto desciende en torno a un 60%, con lo que se puede asegurar que 4 de cada 10 veces el vehículo pasa la curva. Este porcentaje es muy bajo y

en la búsqueda de soluciones para mejorarlo se implementa el paradigma deliberativo en el proyecto y algunas mejoras más para intentar aumentar este porcentaje.

5.3.- PRUEBAS PROTOTIPO HÍBRIDO

Ya utilizando el prototipo híbrido, además de incluir las mejoras que el paradigma deliberativo ofrece, se han implementado otras ya comentadas en el apartado de implementación como el grado de giro variable según el ángulo formado por la línea y una horizontal, la utilización de las dos webcams disponibles, etc.

Con todas las mejoras implementadas se ha intentado incrementar los porcentajes de efectividad que el paradigma reactivo presentaba y desde la primera ejecución con el paradigma híbrido implementado se noto la mejoría y el gran cambio que supone recopilar información para posteriormente decidir qué acción realizar. La estructura de las pruebas será la misma que la presentada en el apartado anterior para poder realizar una comparativa directa entre ambos paradigmas.

5.3.1.- CIRCULANDO EN LÍNEA RECTA

CASO 1

Como en el prototipo reactivo, se ha comprobado el comportamiento del vehículo en condiciones de línea recta sin complicaciones añadidas como puede ser la interferencia de otras marcas viales y demás.

Como se puede observar en la imagen (véase Figura 39), se muestra el vehículo circulando por un tramo recto de la ciudad dentro de los límites del carril y sin ningún problema. Ya con el paradigma reactivo esta situación era bien llevada por lo que no hay razón para dudar de la capacidad del paradigma deliberativo en estos casos y así se demuestra.

En las imágenes obtenidas por las webcam se puede observar las perspectivas distintas y los datos obtenidos en cada una de ellas. Ambas cámaras capturan de manera correcta las líneas que delimita el carril.

A su vez, se ve que en esta prueba el hardware gráfico del sistema ha ofrecido un buen rendimiento ofreciendo 29 *frames* por segundo, un número idóneo para la realización de la prueba.

Figura 39: caso 1: línea recta estándar con prototipo híbrido

CASO 2

En este segundo caso se pone a prueba al vehículo en casos especiales como pasos de cebra y carriles adicionales en línea recta.

Como se puede ver en la imagen de la siguiente página (véase Figura 40) el vehículo se encuentra ante un paso de cebra, algo que en el paradigma reactivo generaba problemas debido a su precipitada decisión.

Las capturas realizadas por las webcam nos muestran dos perspectivas y sin embargo las imágenes tratadas devuelven lo mismo, no se genera confusión y las líneas que se capturan son finalmente las líneas que delimitan el carril obteniendo por lo tanto datos correctos. En caso de no disponer de esas líneas y haber obtenido otros puntos distintos el sistema todavía no habría reaccionado, sino que habría esperado a obtener más muestras que continuaran el progreso que llevan las primeras. En caso de no variar la situación entonces el sistema sí que ejecutaría la orden de moverse.

Además de evidenciar con las imágenes una mejor búsqueda de datos se puede observar que el número de imágenes por segundo ofrecidas por el hardware gráfico no es todo lo bueno que se desearía y sin embargo en este caso, el sistema reacciona de manera correcta ante esta situación y continúa su trayectoria.

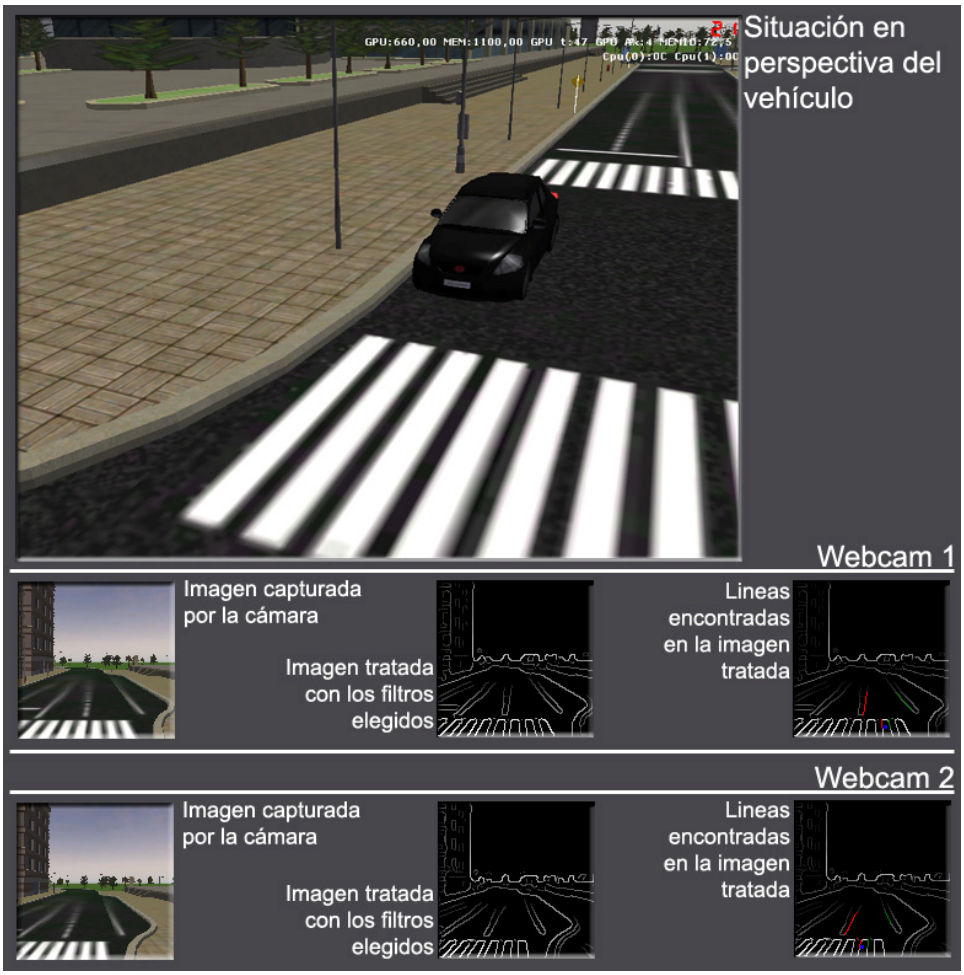


Figura 40: caso 2: confusión con señales viales en paradigma híbrido

La tendencia general de este prototipo es hacer caso omiso a las marcas viales, que no sean las guías que se busca en cada imagen, aunque no queda exento de encontrar un patrón por error durante el suficiente tiempo como para detectar que el vehículo debe moverse. En ese caso la ejecución de dicha acción sería completamente errónea pero, como se ha comentado, esto sucede en casos muy raros y específicos que simbolizan menos del 5% de oportunidades.

De una manera algo más frecuente, encuentra pixeles cercanos al punto central e intenta rectificar la trayectoria con los pequeños cambios de potencia que ya se incluían en el anterior prototipo. Aún así estos pequeños cambios no desenlazan en un final catastrófico ya que son cambios pequeños y delicados que están hechos para colocar el coche en el carril con lo que no provocaría una salida de la calzada.

CASO 3

En este último caso de línea recta se observarán los cambios de potencia para colocar el coche en el carril.

Para este caso no se han tomado imágenes de prueba puesto que la situación se resuelve de igual manera que en el prototipo reactivo. La respuesta por parte del sistema es exactamente la misma ya que el sistema, cada vez que ordena que el vehículo siga recto, comprueba las distancias entre el centro y las líneas que delimitan el carril. Como en el prototipo reactivo, si se detecta que se está demasiado cerca de alguna de ellas se reacciona con la acción correspondiente.

En este caso, no se hace uso del paradigma deliberativo para la realización de esta acción, puesto que no es necesario obtener un determinado número de datos dado que la decisión que se va a tomar no es de mayor transcendencia para la solución final. Los pequeños cambios de dirección no son decisivos para el vehículo y por lo tanto no se ve necesario almacenar datos sobre ellos, es más, incluso entorpecería el buen desarrollo de este recurso retrasándolo en el tiempo y haciendo que el coche se tenga una mayor desviación respecto al carril que debe seguir.

CONCLUSIONES

Teniendo en cuenta los casos estudiados y viendo los puntos donde el prototipo reactivo presentaba una mayor debilidad se puede afirmar con rotundidad que el sistema híbrido para la circulación en línea recta se ha mostrado mucho más efectivo demostrando una efectividad del 90%. El 10% queda reservado para casos especiales en los que no es capaz de gestionar las marcas viales y toma decisiones erróneas.

Desde luego el porcentaje obtenido demuestra una gran estabilidad los que nos indica que el vehículo sería capaz de circular prácticamente sin problemas durante un largo perdido de tiempo en línea recta.

A su vez, el mantener los pequeños cambios de dirección para colocar el vehículo en recta dentro del paradigma reactivo ha sido un acierto, ya que ese tipo de reacciones es necesario tenerlas disponibles en cualquier momento como contramedida a una mala colocación del vehículo.

5.3.2.- SEMÁFORO

Para los casos en los que nos encontramos ante un semáforo se ha de indicar que el sistema no ha cambiado y que para estos casos, debido a la velocidad de reacción que se necesita, se ha mantenido el paradigma reactivo.

Por ello las pruebas mostradas y las conclusiones obtenidas en el apartado correspondiente a las pruebas del prototipo reactivo son válidas para este apartado. Además, la efectividad obtenida por parte del prototipo reactivo fue muy buena y no es necesario realizar cambios que afectasen a esta efectividad aunque sí que se probó a integrar esta decisión en el paradigma deliberativo.

Los resultados de esta inclusión fueron buenos pero no tanto como los obtenidos con el prototipo reactivo. Se observaba que en una situación en la cual el semáforo se acaba de cambiar, el vehículo se detiene tarde y se salta el semáforo dejando fuera del alcance de las cámaras el semáforo justo cuando el vehículo se detiene por completo. En ese instante, de nuevo comienza la deliberación y al no encontrar semáforo el vehículo avanza por el cruce estando en rojo, algo mucho más peligroso que quedarse parado encima del paso de cebra aunque moleste a los viandantes. Por estas razones se decidió dejar el paradigma reactivo para los semáforos.

5.3.3.- GIRANDO

En este apartado, al igual que en los anteriores, se va a probar el funcionamiento del sistema cuando el vehículo afronta una curva y las posibles situaciones que deriven de ello.

CASO 1

En esta prueba se comprobará cómo reacciona el sistema ante una curva normal y corriente sin ningún tipo de peculiaridad ni dificultad. Hay que recordar que para el giro en curva se han

implementado varias mejoras con respecto al prototipo reactivo, además de incluir el paradigma deliberativo.

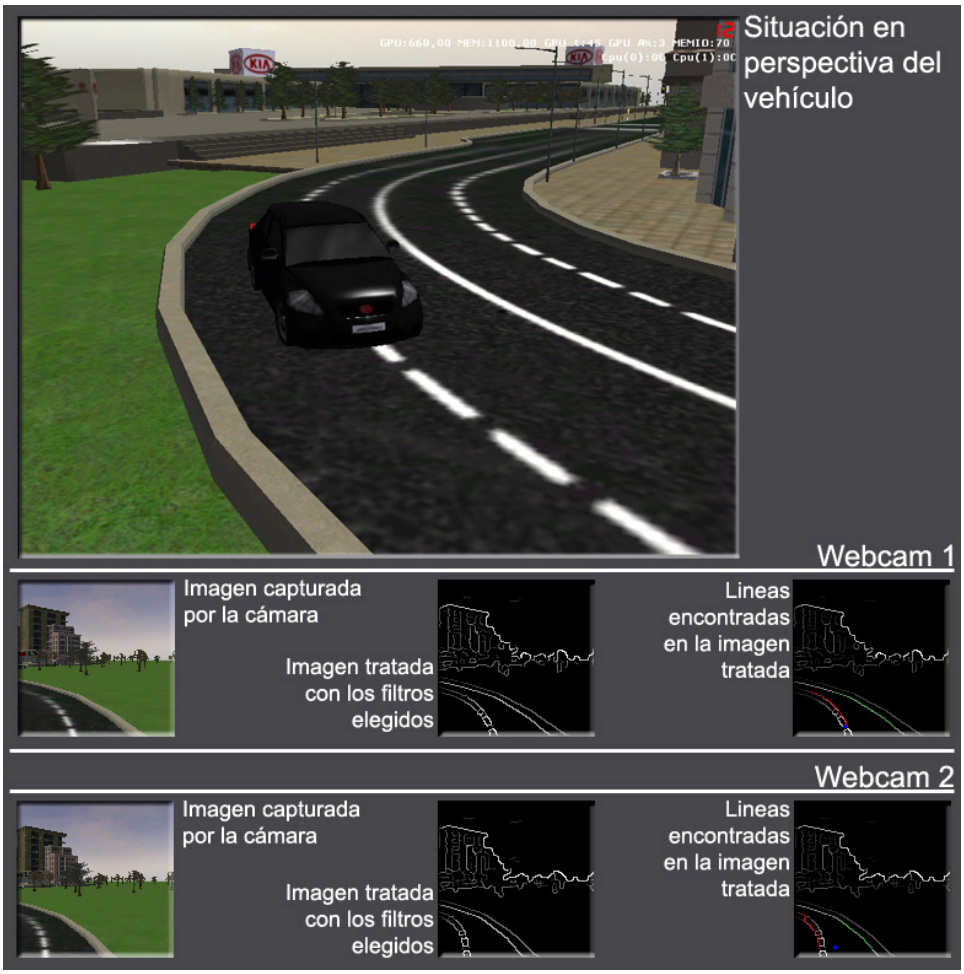


Figura 41: caso 1: giro correcto en prototipo híbrido

Como se puede apreciar en la imagen superior, se ve al vehículo girando de manera correcta, invadiendo un poco el carril contiguo pero dado la estrechez de los mismos no se considera como una penalización. En las imágenes capturadas por al webcam se puede apreciar la diferencia de perspectiva de ambas webcams, algo que se ve más claramente cuando se observa las líneas halladas por el sistema de búsqueda de información.

Gracias al paradigma deliberativo introducido en estos casos de giro se ha logrado realizar un giro más estable y, principalmente, girar cuando el sistema delibere que está ante una curva. Con esta decisión se logra girar en el momento en que la carretera marca el giro, ni antes, ni después, lo que otorga más fiabilidad a la hora de trazar las curvas.

Además, se puede observar en la imagen superior cómo el hardware gráfico no ha ofrecido todo el potencial que se desearía y aún así el sistema ha sido capaz de reaccionar a tiempo para no salirse de la calzada. En otros casos, es posible que debido a este hecho el vehículo no reaccione con la suficiente antelación y, aunque comience a girar, al final se acabe saliendo de la calzada.

CASO 2

En este caso se estudiará el por qué el vehículo con un sistema híbrido implementado llega a salirse de la calzada.

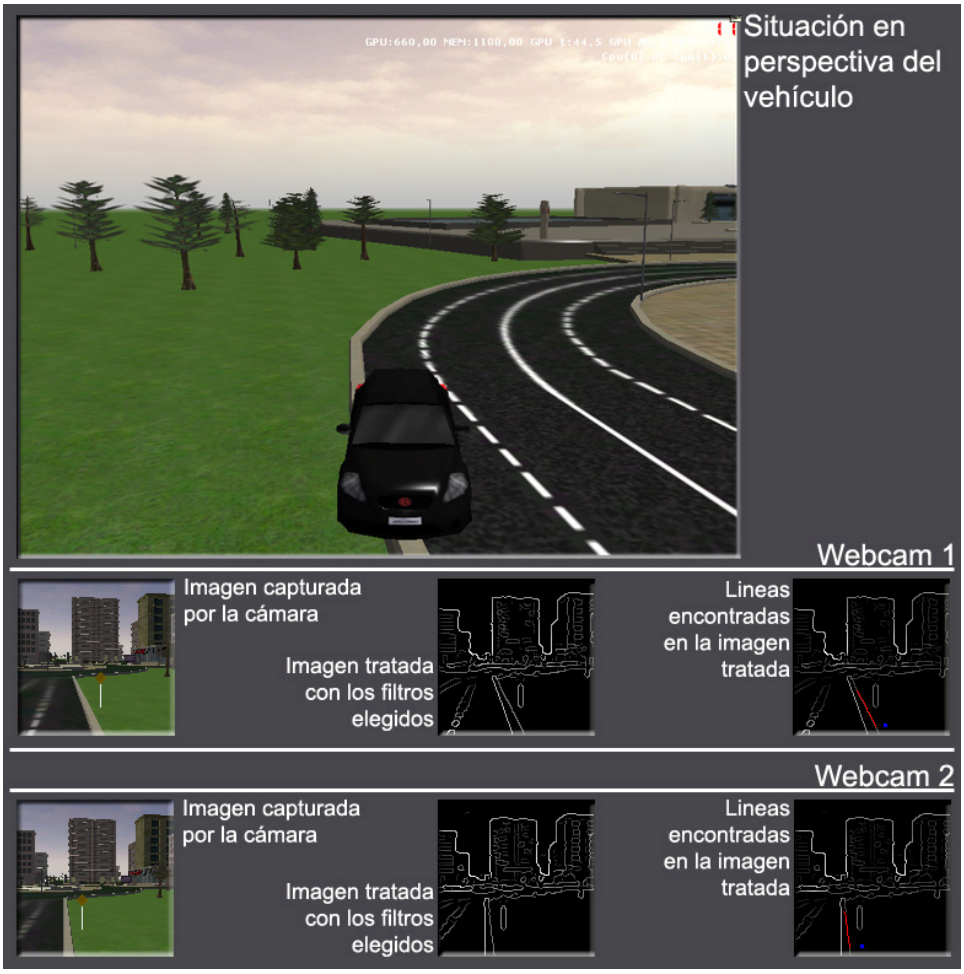


Figura 42: caso 2: salida inminente en prototipo híbrido

Como se observa en la imagen superior, se está ante una situación de salida inminente de la calzada. En este caso el vehículo llevaba una trayectoria muy justa respecto a los límites exteriores de la curva hasta que finalmente se salió de la calzada y, una vez en el exterior, las posibilidades de recuperar una posición válida son muy bajas.

Esta situación es provocada por varios factores. El más importante es que el sistema pierde una de las líneas de referencia lo que hace más complicado decidir qué acción se ha de realizar. Como se observa en las imágenes tratadas, se puede observar que la única línea de referencia para el sistema es la línea pintada de rojo, cuando esa línea debería ser la exterior y no la interior como se está interpretando en estos momentos. Otro factor se puede observar en la figura grande y es que el pésimo rendimiento del hardware gráfico, que tan solo nos ofrece 11 *frames* por segundo hace que el vehículo comience a girar demasiado tarde y por lo tanto que vaya muy ceñido al borde exterior lo que finalmente provoca esta pérdida de líneas de referencia y por lo tanto la salida del vehículo de la calzada.

Esta situación, en condiciones optimas, es difícil que se produzca ya que el sistema por lo general tiende a reaccionar momentos antes o incluso en el mismo momento en que la curva comienza a mostrar curvatura con lo que el sistema se muestra muy estable en dichos casos, que son representados por el caso 1 de este apartado.

CASO 3

En esta última prueba se comprobará los posibles cambios de carril involuntarios que el vehículo pueda realizar. Estos cambios de carril se podrían diferenciar entre los tolerables y los no tolerables. En el primer grupo entrarían los cambios al carril contiguo lo cual es erróneo pero no fatal para una hipotética situación real y por otro lado estaría la incursión del vehículo en el carril o carriles de sentido contrario, algo más grave aunque se da con menos frecuencia.

Como se puede observar en la imagen (véase Figura 43), el vehículo ha invadido por completo el carril contiguo al que le corresponde. A partir de este momento, regresar al carril original será una tarea complicada puesto que el sistema de referencia de líneas ha cambiado las líneas a las que delimitan el carril en el que el vehículo se encuentra actualmente.

Una de las posibilidades para lograr regresar al carril original podría surgir cuando ocurre un cambio de estado en la carretera, es decir, que el estado de la misma cambie a recto o nueva curva pero en otra dirección, en ese caso y dependiendo del azar podría volver el vehículo a su carril original mediante las correcciones en línea recta u otros recursos. Aún así, será difícil que el vehículo invada el carril contrario aunque hay posibilidades de ello. Estas posibilidades son realmente bajas ya que, para invadir el carril contrario en la carretera suministrada por RUC es necesario invadir el carril contiguo y posteriormente invadir el contrato.

Otro de las factores que ha podido generar esta situación es el bajo rendimiento que se ha obtenido en esta prueba por parte del sistema hardware, que en este caso solo ha sido capaz de ofrecernos 8 *frames* por segundo lo que puede generar lentitud en las decisiones provocar que el vehículo se mantenga con un grado de giro erróneo más tiempo del necesario y por lo tanto invadir el carril contigo y hacer que las referencias cambien para el sistema.

CONCLUSIONES

En el prototipo reactivo, cuando se hablaba de giros se observaba que el gran problema de dicho paradigma era que el vehículo se salía demasiado de la calzada. Con la ayuda de estas pruebas es claramente observable que la fiabilidad del conjunto se ha aumentado enormemente hasta tal punto que este prototipo ha logrado aumentar el porcentaje de estancia dentro de la calzada, teniendo en cuenta invasiones de carril, algo menos grave que salirse de la carretera.

Figura 43: caso 3: invasión de carril en prototipo híbrido

Esto se debe a varios factores, uno de ellos es la obtención de más información y sobre todo la toma de decisiones teniendo en cuenta dicha información. Cuando se toma la decisión en este prototipo se puede afirmar, con un 90% de posibilidades de acierto, que el vehículo se encuentra ante una curva y no solo eso, sino que el mismo sistema es capaz de detectar con qué grado de giro debe girar el vehículo, es decir, es capaz de interpretar la curva y detectar cómo de cerrada o abierta es.

Esta mejoría se simboliza en números, números que confirman que la fiabilidad en el giro se ha aumentado enormemente y de manera contundente pues que tan solo 1 de cada 10 veces el vehículo se sale de la calzada.

5.3.4.- ROTONDAS

En este apartado, solo aplicable a los prototipos híbridos, se muestra la efectividad del sistema para afrontar las rotondas que pueda encontrar el vehículo en su camino.

CASO 1

En este caso se estudiará cómo actúa el vehículo ante una rotonda, encontrándose este en una situación óptima para tomarla. Se ha de hacer hincapié en que el sistema implementado para estas situaciones no es el más eficaz dado que se simplemente se ha limitado a realizar un sistema que trazara las rotondas y que nos permitiera seguir circulando por los carriles sin entrar en mayores complicaciones.

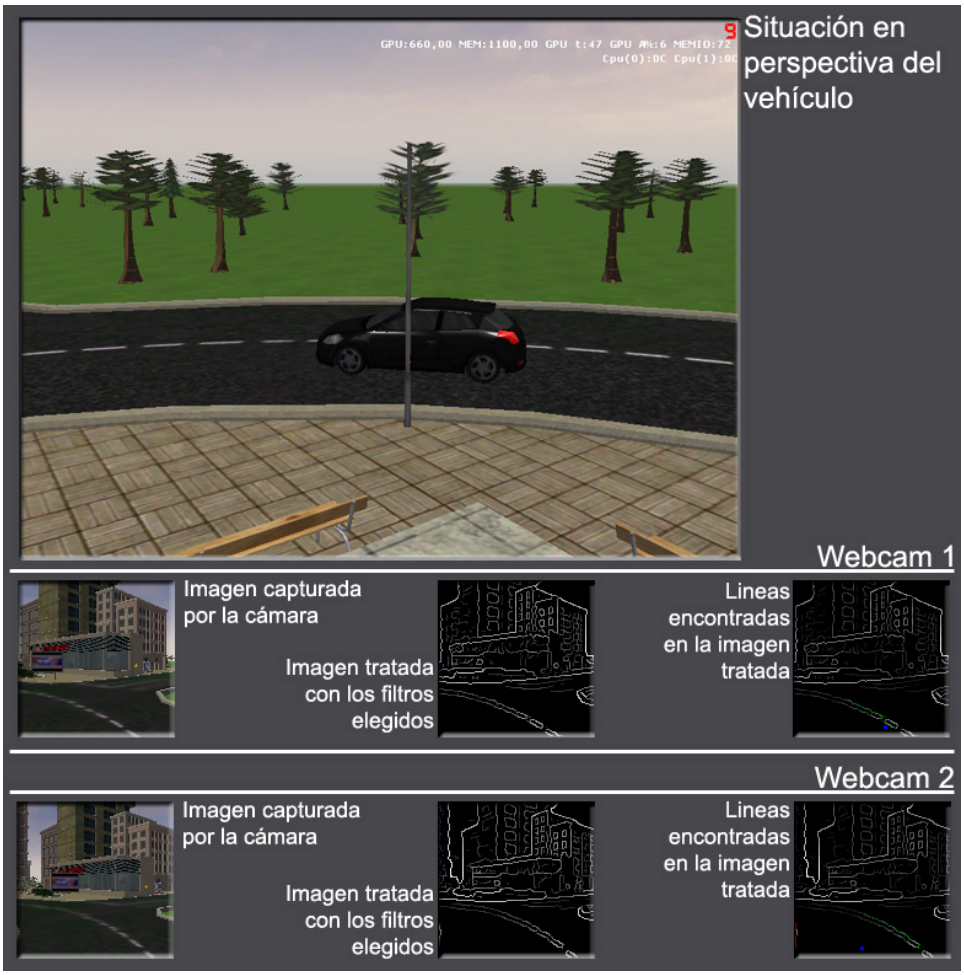


Figura 44: caso 1: rotonda tomada correctamente.

Como vemos en la imagen superior, se observa que el vehículo cruza los dos carriles de la rotonda y que su destino no es otro que la primera salida a la derecha. En este caso las imágenes obtenidas por las cámaras tan solo comunican cuando se ha salido de la rotonda ya que cuando el vehículo se encuentra dentro de la misma todas las líneas obtenidas serán obviadas y hasta que no se encuentre un indicativo de que se ha salido de la misma no se vuelven a tener en cuenta dichas líneas.

Los datos que se han de obtener para dar por finalizada la rotonda deben ser líneas que indiquen que se debe girar a la derecha ya que el vehículo cuando se introduzca en el carril, por el cual debe salir de la rotonda, no llegará recto completamente sino que entrará cruzado detectando líneas que le indican un giro a la derecha. Con este indicativo, el vehículo se coloca sobre el carril que ha detectado y continuaría su marcha dando por finalizada la rotonda.

Como también se puede observar en la imagen grande, en esta zona del mapa dado por RUC se obtiene un bajo rendimiento del sistema hardware aunque, como se ha comentado, una vez dentro de la rotonda la respuesta de las imágenes no es tan necesaria, sin embargo, si es importante la salida de la rotonda y la decisión de entrada a la misma puesto que una decisión pronta o tardía puede decidir el buen destino del paso de la rotonda.

CASO 2

En este segundo caso se analizará las rotondas realizadas por el vehículo cuando no se encuentra en una posición óptima o cuando se decide un poco tarde cuando girar.

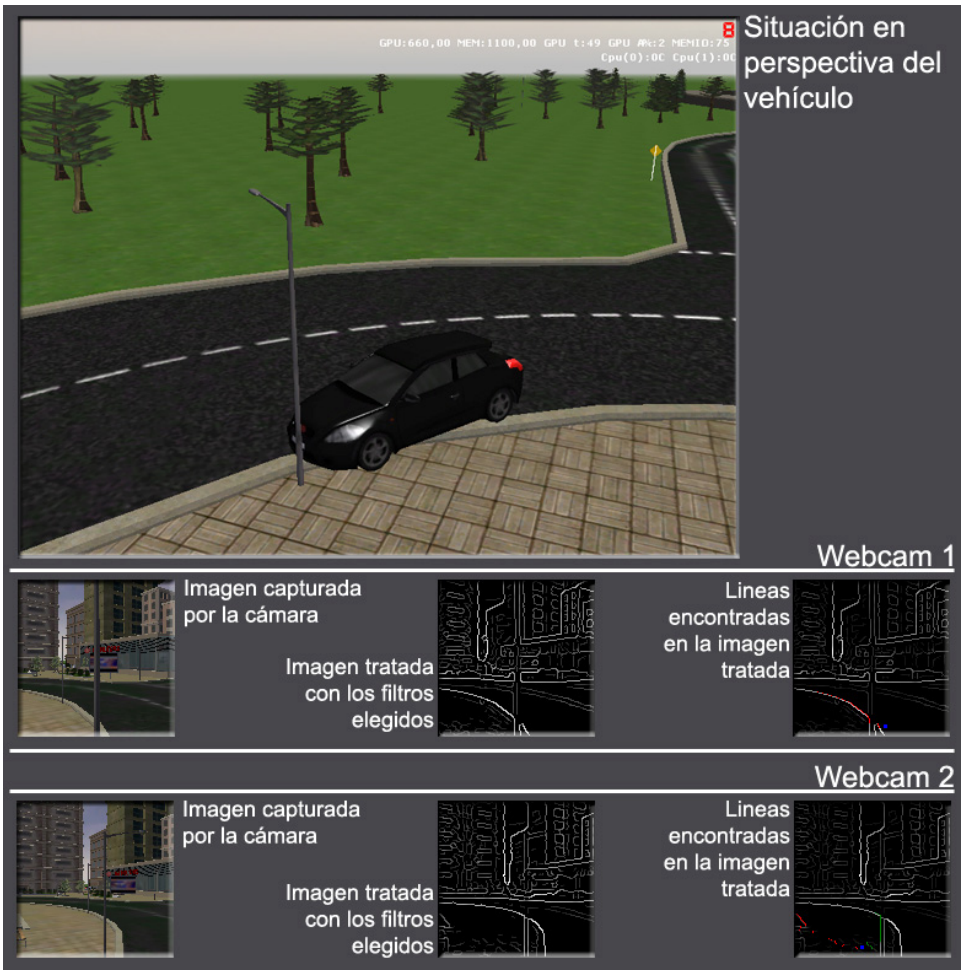


Figura 45: caso 2: rotonda errónea.

Como se puede observar en la imagen grande se ve como el vehículo no ha girado lo suficientemente pronto y ha terminado impactando con la farola. Como se ha comentado en el apartado de implementación, el sistema desarrollado para el tratamiento de rotondas es simple y llano para lograr alcanzar los objetivos de tiempo marcados en la realización del proyecto. Por ello solo se dispone de un grado de giro fijo para pasar la rotonda. En caso de que el vehículo gire un poco antes un poco después es posible que no se sobrepase satisfactoriamente.

Además de lo comentado, se observa que el rendimiento del sistema hardware no ha sido del todo el más eficaz por lo que puede haber provocado que el giro se realizara tarde con lo que el vehículo ha ido directo hacia la farola.

CONCLUSIONES

Observados los comportamientos del vehículo en las rotondas, se puede concluir que el sistema es mejorable en multitud de aspectos, aunque como se ha comentado con anterioridad, la solución actual se ha implementado para delimitar el proyecto y a la vez permitir que el vehículo continúe su marcha en un número comprensible de veces.

Este número comprensible se ha intentado aproximar a la mitad de las veces, aunque dependiendo de la situación puede variar este número.

5.4.- CONCLUSIONES GENERALES

En este apartado se demostrará y justificará la elección del paradigma híbrido a la hora de realizar el sistema. Para ello, este apartado tomará como fuente de información fiable las pruebas realizadas y comparará de manera directa ambos prototipos.

5.4.1.- RESULTADOS EN GIROS

Antes de llegar a la conclusión final analizaremos una de las partes más importantes del sistema y prácticamente la única parte que tiene considerable dificultad que es el giro del vehículo. Ir en línea recta pude ser sencillo en casos en los que el vehículo ya esté centrado sin embargo los giros son la parte complicada de este proyecto.

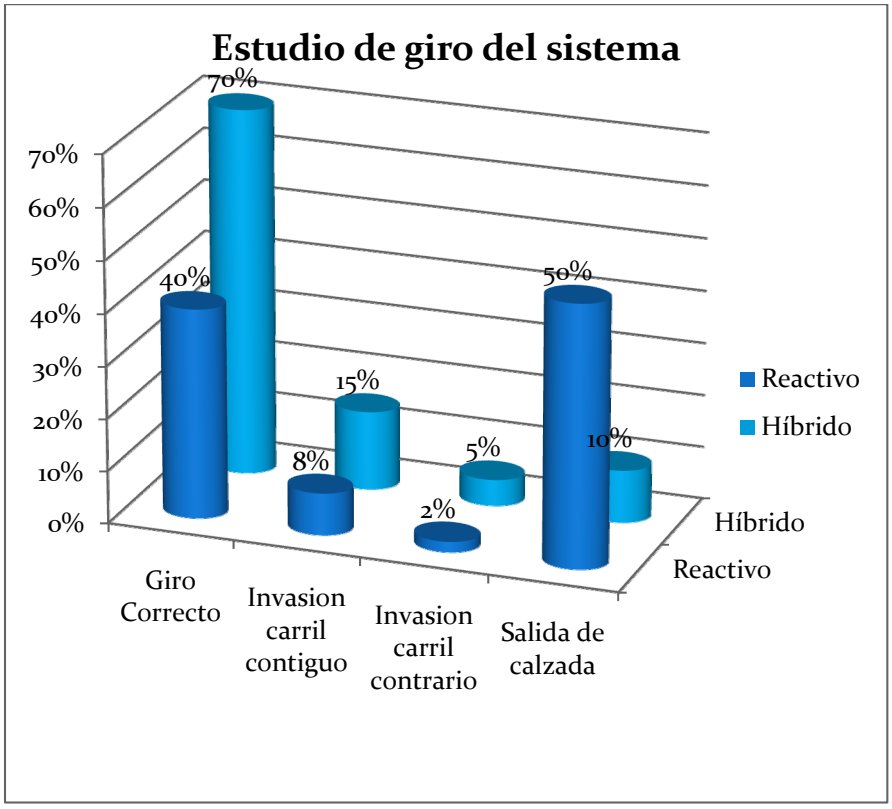


Figura 46: gráfica comparativa de efectividad en el giro

Como se demuestra en la gráfica, el sistema híbrido es capaz de girar correctamente en un mayor porcentaje que el sistema reactivo. Esto es algo lógico ya que el sistema híbrido es capaz de obtener más datos para la realización del giro y recopila más información que posteriormente tomará en cuenta para deliberar la acción que ha de ejecutar el vehículo. Esto sumado a numerosas mejoras implementadas hace que el sistema híbrido sea el más acertado para esta situación.

También se puede observar en la gráfica como el sistema reactivo se sale más veces de la calzada en comparación con la invasión de otros carriles, algo más grave ya que en esa situación el vehículo es irrecuperable. Sin embargo, y aunque no sea la mejor situación, el prototipo híbrido invade más veces los carriles contiguos o contrarios. Invadir el carril contiguo se podría considerar como un fallo menor ya que seguimos cumpliendo las normas de la vía siempre y cuando sea de dos carriles. Algo más grave es

invadir el sentido contrario, pero en ambos casos el vehículo sigue en la calzada y con un sistema de referencia que le guía aunque sea en un lugar incorrecto.

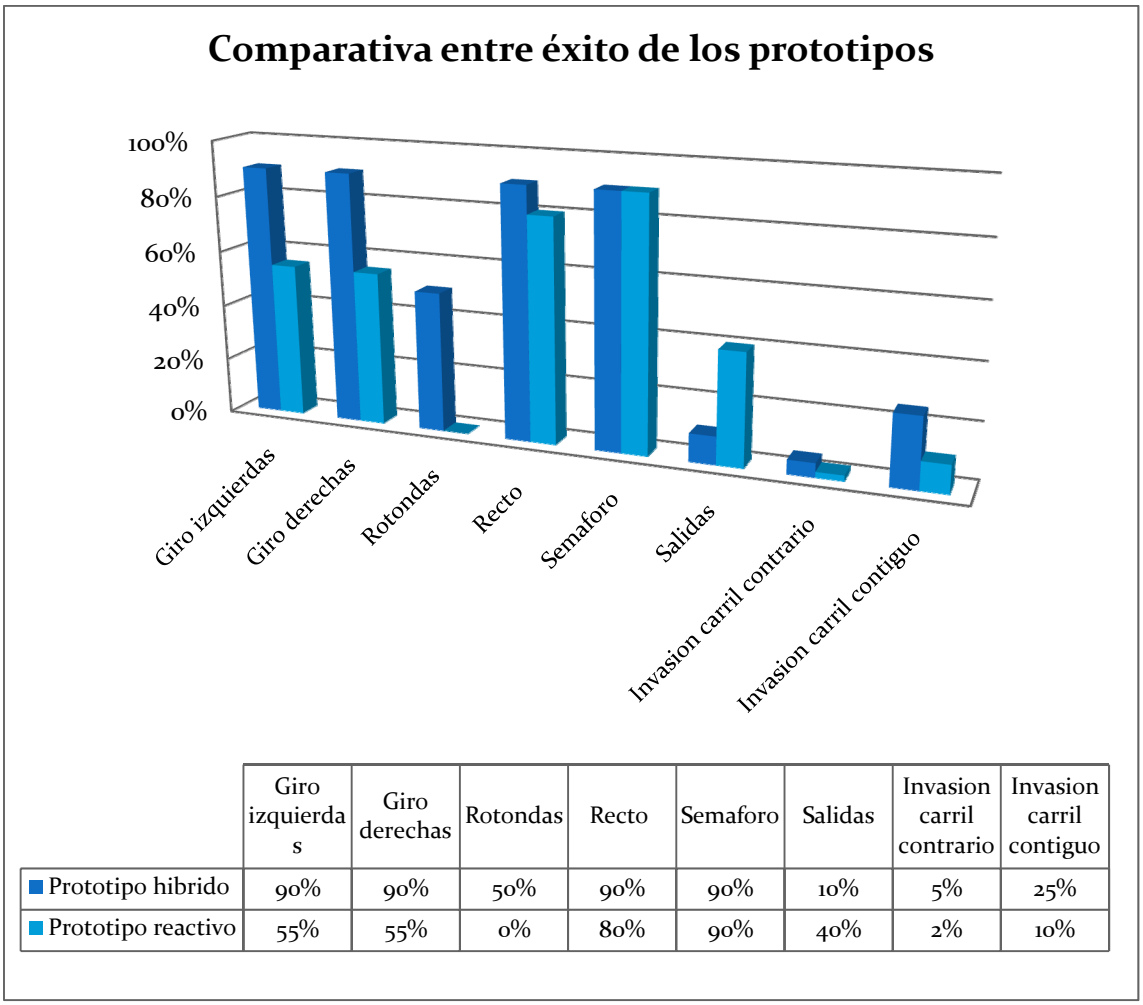
Con todos estos datos se puede afirmar con total rotundidad que el sistema híbrido es el acertado para situaciones de giro. Además, se entiende que se ha logrado unos porcentajes bastante decentes de cara a la realización de los giros y más teniendo en cuenta el sistema tan delicado. Obviamente perseguir el 100% de efectividad es la gran meta pero a su vez es un valor muy difícil de obtener debido a la cantidad de variables a manejar y la variedad de situaciones en las que el vehículo se puede encontrar.

5.4.2.- RESULTADOS GENERALES

Una vez estudiado los resultados en la situación más delicada que el vehículo puede encontrarse, se procede a estudiar de manera general la efectividad del conjunto.

En general y como se comprobará más adelante con algunas gráficas, la efectividad del conjunto ha mejorado con la inclusión de paradigma híbrido al prototipo reactivo. Con el prototipo reactivo muchas reacciones eran precipitadas lo que hacía que el vehículo intentase rectificar esas decisiones precipitadas con actuaciones que hacían que el vehículo terminara fuera de la calzada. Mediante la toma de datos y una posterior deliberación se espera el tiempo necesario para tomar la decisión hasta que el sistema no es té totalmente seguro de la decisión que debe tomar.

En cuanto a consecución de objetivos, se puede afirmar que el sistema autónomo creado cumple con los objetivos especificados y que sus resultados se encuentran dentro de unos números aceptables para su ejecución. Naturalmente, estos números son mejorables con técnicas más avanzadas, algunas de ellas explicadas en el siguiente punto, pero debido al ámbito en el que nos encontramos, un proyecto fin de carrera, no se ha creído conveniente continuar con el desarrollo y aumentar su efectividad dado que el proyecto debe tener una duración coherente con el contenido que se ha de desarrollar.



	Giro izquierdas	Giro derechas	Rotondas	Recto	Semaforo	Salidas	Invasion carril contrario	Invasion carril contiguo
Prototipo hibrido	90%	90%	50%	90%	90%	10%	5%	25%
Prototipo reactivo	55%	55%	0%	80%	90%	40%	2%	10%

Figura 47: gráfica comparativa de resultados generales

Como se puede ver en el gráfico (véase Figura 47) los números que obtiene el prototipo híbrido son más positivos en las acciones que requieren éxito y a su vez reduce el porcentaje de salidas de la calzada. Si bien, es cierto que el porcentaje de incursiones en otros carriles se aumenta con el paradigma híbrido se considera más correcto invadir el carril contiguo que salirse de la calzada perdiendo la posibilidad de volver a ella.

También es observable cómo el apartado de semáforo mantiene la misma efectividad debido a que se mantiene el paradigma reactivo cuando nos encontramos en dichos casos.

En el apartado de rectas se ha incluido esa ligera mejoría debido a que en un mayor porcentaje el vehículo no se confunde con carriles adicionas o marcas viales en el asfalto que pudieran confundir al sistema de búsqueda de datos en la imagen.

Uno de los apartados que queda más por evolucionar es el paso por rotondas del sistema. Nuevamente, como se ha comentado antes, debido al ámbito de este trabajo se ha centrado los esfuerzos en una circulación sin rotondas ni cruces y se ha hecho un sistema sencillo mediante el cual se pueda seguir circulando aunque se sabe que no es el más correcto y que se podría mejorar.

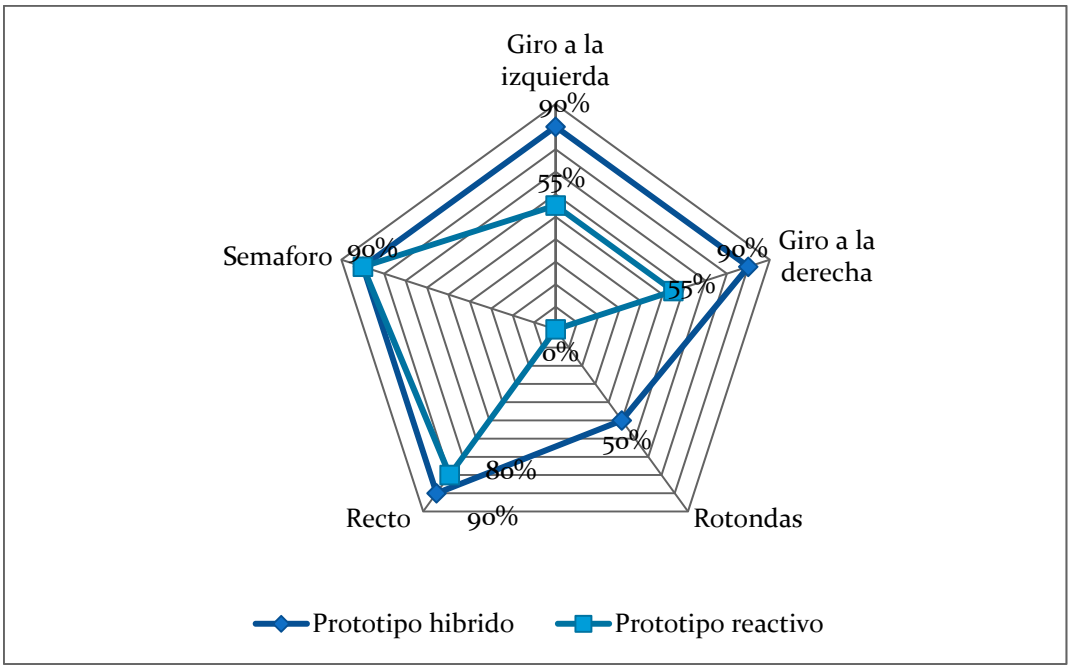


Figura 48: diagrama radial de éxito entre prototipos

En esta gráfica se puede comprobar de manera muy visual las diferencias y la completitud a la que llega el sistema híbrido. Además muestra que el proyecto se podría afinar más en las rotondas ya que se podría decir que es el único aspecto que necesita una mejora. El resto de aspectos podrían ser más eficientes pero no se considera necesario o imprescindible.

6.- CONCLUSIONES Y LÍNEAS FUTURAS DE TRABAJO

Tras el estudio de la tecnología aplicada y la realización de este proyecto, se está ante un sin fin de posibilidades y un trabajo que podría ser amplísimo. Aunque primero, se tratarán las conclusiones sacadas de este proyecto y, posteriormente, se comentarán las posibles ampliaciones que podrían realizarse sobre el mismo en un futuro.

6.1.- CONCLUSIONES

Una vez finalizado el proyecto, una de las mayores conclusiones que se pueden obtener es el gran aprendizaje obtenido del uso de arquitecturas robóticas. Se ha logrado comprobar de manera práctica cómo una arquitectura cien por cien reactiva para trabajos que requieren deliberación y la toma de una decisión concreta no sirve y se hace muy complicado obtener una solución final que satisfaga los objetivos marcados.

En el proyecto, se puede observar cómo, debido a una mala toma de datos o una imagen que muestra datos que no siguen el patrón, el vehículo cambia de dirección perdiendo toda la fiabilidad que tiene. Visto de este modo, incluso se podría considerar un logro llegar a la primera curva aunque en muchos casos el vehículo se queda por el camino.

Por lo tanto, se ha evidenciado que el enfoque reactivo no era el más idóneo para este tipo de problemas y a la vez, comprobando el apartado de pruebas, se ha demostrado que el enfoque híbrido, con la inclusión del apartado deliberativo, es el enfoque más correcto para la situación a la que se enfrenta proyecto. Ya desde su primera implementación, la mejora en la toma de decisión se vio incrementada sustancialmente. Naturalmente, al igual que la mayoría de las cosas que nos rodean, este enfoque tiene sus pros y sus contras como por ejemplo el retraso que tiene el sistema para tomar una decisión debido a que tiene que comprobar varios datos simultáneamente.

Aún así esto se puede solventar mediante contramedidas como por ejemplo detener el vehículo si es necesario. Si no se obtienen datos y no sabemos adónde vamos, pues se está en la obligación de detener el vehículo, obtener datos y cuando esté todo listo, continuar con la decisión tomada. Por suerte, en este proyecto se ha encontrado ese equilibrio con lo que no ha sido necesario detener el vehículo en ninguna situación, si bien, la velocidad máxima que el vehículo alcanza es bastante baja, precisamente para favorecer este hecho.

Otra conclusión que se puede obtener es la similitud de los comportamientos robóticos con los humanos. En numerosas ocasiones, tanto en privado como en reuniones con el tutor, se ha recurrido a la inspiración en los procesos humanos cognitivos para decidir qué hay que hacer. Sobre todo a la hora de realizar giros, en el momento de introducir el giro dinámico, es decir, adaptado según la posición del coche y la curva en la que nos encontramos. En la realidad, es una acción lógica y normal para un humano. Si el humano detecta que debe girar más por que el vehículo se sale de la calzada, este aumenta el grado de giro del volante. Este pensamiento es el que se ha intentando transmitir al sistema y se ha conseguido de manera que cumple los objetivos marcados al inicio.

Este ejemplo comentado es tan solo una de las aplicaciones de razonamiento humano al proyecto, hay otros casos, los cuales no se van a comentar por razones de síntesis, que se han aplicado al robot para lograr conseguir el éxito de este producto.

Por otro lado, cabe destacar la alta dependencia de este proyecto del hardware disponible. MRDS con su MVSL exige muchos recursos, sobre todo gráficos. Para un buen funcionamiento de las webcams es necesario disponer de al menos 30 frames por segundo para que funcionen de manera óptima. Si se consiguieran menos frames, cabe la posibilidad de retrasarse en la toma de las decisiones o de perder fiabilidad. Este hecho hace que se deba trabajar de una manera óptima, ahorrando recursos y dedicándolos todos al MRDS. El ordenador utilizado para la realización del proyecto ha sido un Intel Pentium 2.4ghz Dual Core con una tarjeta gráfica Asus Ati EAH4850 de 512 megabytes de memoria gráfica. Quizás el equilibrio en

este ordenador entre procesador y gráfica no sea el más adecuado, ya que la gráfica es más potente que el procesador, si asemejamos ambos componentes a su nivel. Esto presentó algunas dificultades añadidas ya que el comportamiento del proyecto era algo variable dependiendo de los recursos disponibles en ese momento. Aún así se intentó realizar el proyecto para funcionar en todas las condiciones posibles y que se adaptara a todas las dificultades que se pueden encontrar.

Es importante comentar el proceso de aprendizaje de MRDS. Si bien no es muy complicado, ya que posee un fácil aprendizaje, la complejidad reside en la multitud de elementos totalmente nuevos que surgen de la utilización de MRDS, como por ejemplo la concurrencia, servicios, puertos, subscripciones a servicios, términos nunca vistos antes y que requieren de un proceso de adaptación. En este aspecto, los tutoriales proporcionados por MRDS y el seguimiento de los mismos en la página web de Microsoft han sido de grandísima utilidad para conseguir un conocimiento básico con el cual poder entender el funcionamiento de esta nueva tecnología. Aún así, estos conocimientos son limitados y es cierto que se podría estudiar la materia más en profundidad pero era algo que no se consideraba necesario para la realización de este proyecto, aunque es para tomarlo en consideración para un enriquecimiento futuro.

Entrando en el código, se ha implementado de manera que solo utiliza los recursos ofrecidos por RUC pero no modifica nada del código base del RUC, por lo tanto sería posible llevarse el servicio del cronometro e implementarlo en otro proyecto que contenga webcams. Esto hace que sea muy modular y adaptable, de hecho sería posible llevarlo a un coche real y hacerlo funcionar pero evidentemente estamos hablando de un gran gasto de recursos que para la realización de este proyecto poco sentido tenía. Se ha intentado realizar todo lo modular posible, incluso es posible modificar el numero de muestras tomadas con el simple cambio de dos valores. Tan solo cambiando esos dos valores el código se adapta por completo, las decisiones y demás componentes están programados de tal manera que se adaptarían a este cambio sin necesidad de modificar más cosas. Un punto un poco menos flexible es el estudio de las imágenes en las que se ha centrado pensando en que las imágenes disponibles serían de 128x128. En caso de cambiar la resolución de estas imágenes el código se debería cambiar pero se ha de tener en cuenta que sería cambio de grandes dimensiones ya que se cambia por completo la fuente de información que tenemos, algo de vital importancia para el desarrollo del proyecto.

Las ventajas que se pueden obtener de este proyecto es que con pocos recursos se ha logrado implementar un sistema de visión artificial para conducir un vehículo sin ningún tipo de asistencia. Aunque el sistema, como indican las pruebas, en ningún momento se muestra infalible, si muestra seguridad distintos momentos clave, como puede ser a la hora de girar. Otra de las ventajas de este proyecto es la oportunidad de realizar una gran cantidad de pruebas sin la necesidad de poner en juego un coche real o un robot autónomo gracias al entorno virtual que se suministra, incluso no solo el mismo vehículo, sino las cámaras y demás elementos hardware los cuales incrementarían el coste total del proyecto.

Como ya se ha introducido en un párrafo anterior, los servicios desarrollados son reutilizables en cualquier momento en otro proyecto de MRDS ya que pueden ser reutilizable siempre que estemos dentro del mismo marco, marco definido por el MRDS.

Por el contrario y como inconveniente, el hecho de desarrollar este proyecto en un entorno virtual hace que migrarlo a un mundo real sea una tarea difícil, sobre todo por las imágenes que serán capturadas en el mundo real serán mucho menos nítidas y con más ruido que entorpecería en gran medida la búsqueda de las líneas. Además, otro de los inconvenientes encontrados es el excesivo consumo de recursos y la obligatoriedad de dispones de una tarjeta gráfica potente para poder desarrollar el entorno virtual en el que el proyecto se desarrolla.

Como conclusión final se puede afirmar que el proyecto ha cumplido los objetivos que se diseñaron antes de comenzar con el desarrollo del mismo. Se ha conseguido un sistema autónomo capaz de mover un vehículo por una ciudad con la ayuda, únicamente, de sus elementos hardware y de la información proporcionada por el entorno.

6.2.- LÍNEAS FUTURAS DE TRABAJO

Dentro de este apartado, hay multitud de posibles líneas futuras debido a la gran amplitud de este proyecto y las grandes posibilidades que ofrece la robótica actual. Teniendo en cuenta que se ha recortado el ámbito del proyecto muchísimo debido a la gran cantidad de posibilidades y a la dificultad de realización, en tiempo, de todas ellas.

PASO POR ROTONDAS

Una línea futura de trabajo y a la vez sencilla es una mejor gestión del paso por rotondas, detectando la rotonda en sí y las posibles salidas que tiene la misma. Como se recuerda, el paso por rotonda del proyecto mostrado es un giro fijo constante hasta salir por la primera salida disponible. En caso de realizar esta mejora se podría incluso decidir por qué salida ir o simplemente dar vueltas por la rotonda. Además, esta mejora no provocaría grandes cambios de código, simplemente realizar una sección donde se gestione el paso por rotondas y posteriormente agregarlo a la toma de decisiones como si de una decisión más se tratara. Otra opción que se podría aplicar es la utilización de una capa de control a nivel más alto para la toma de decisiones para posteriormente ejecutar las órdenes correspondientes.

MEJORA GESTIÓN DE CRUCES

En un futuro, se podría realizar un sistema que fuera capaz de gestionar cruces, ya que el actual sistema se centra en la circulación por una calle continúa. Dicho sistema ayudaría a poder continuar la marcha del vehículo tanto por el carril actual como por otra calle. Para ello se podría hacer estos cambios de dirección aleatoriamente o mediante una navegación global, algo que ya se sale del ámbito del proyecto.

INTEGRACIÓN DEL GPS

Otra línea de trabajo, esta ya algo más compleja, es la utilización del GPS integrado en el vehículo y que RUC pone a nuestra disposición en el *starter kit.* Con la inclusión del GPS se podría almacenar el camino recorrido para no volver a pasar por él y así obtener de una manera más efectiva un recorrido perfecto. Naturalmente, se está teniendo en cuenta que no se dispone del mapa, tan solo de las coordenadas que el vehículo tiene al moverse. Con la utilización del GPS se podría llegar a niveles de aprendizaje nunca visto ya que incluso el sistema podría aprender rotondas y almacenarlas y repetir ese comportamiento o curvas, giros, cambios de sentido, etc.

Las posibilidades se disparan, por ello quedó fuera del alcance del proyecto ya que podría haberse expandido mucho más. Una vez aprendido el mapa e inspeccionado la zona sería posible conducir con el vehículo de una manera más segura, incluso aumentando la velocidad en zonas rectas y frenando cuando se llegue a una curva, diseño de rutas para ir de un punto “a” a un punto “b” de distintas maneras…Básicamente las funcionalidades que cualquier GPS que se puede adquirir hoy día, siempre y cuando el mapa fuera conocido.

Esta ampliación otorgaría al vehículo una mayor seguridad a la hora de realizar acciones. Naturalmente, esto es algo complicado, puesto que el mapa puede ser de tres kilómetros o de quinientos kilómetros con lo que el aprendizaje de un mapa depende mucho del tamaño del mismo.

NUEVAS TECNOLOGÍAS DE INTELIGENCIA ARTIFICIAL

Otra línea de trabajo podría ser la utilización de otras tecnologías de inteligencia artificial a la hora de tomar las decisiones. Actualmente, en el proyecto mostrado se utilizan unas reglas que vienen a ser condiciones que se comprueban con una base de hechos que no son más que los datos almacenados en las muestras. Con estos hechos y las reglas se toma la decisión. La inteligencia artificial es muy amplia en este sentido y es posible aplicar otras tecnologías inteligentes como las redes neuronales. Mediante la utilización de redes neuronales se conseguiría simular qué comportamiento de la carretera seguirá al actual, es decir, tal y como un humano es capaz de predecir la curva y predecir que tendrá que aplicar un grado determinado de

giro, las redes neuronales serían capaces de simular este razonamiento y predecir mediante patrones conocidos y aplicados anteriormente el comportamiento de la vía. Estas redes con capaces de sustituir en mayor o menor medida el cerebro humano y el razonamiento humano, tanto es así, que incluso reproducen los errores de razonamiento que son inherentes al ser humano.

TAMAÑO DE LAS IMÁGENES

Otra posible vía para evolucionar este proyecto podría ser la utilización de imágenes con una mayor resolución. En el proyecto actual, como se ha comentado, se utilizan imágenes en una resolución bastante pequeña, 128x128 pixeles lo que limita un poco el campo de visión y complica la obtención de datos veraces. Con una imagen un poco más amplia se podría definir un poco más los detalles de cada una de las imágenes aunque, como es obvio, el proceso de gestión de la imagen sería más complicado y lento al manejar muchos más datos. Aún así sería optimo un tamaño de imagen un poco más amplio de mínimo 256x256 pixeles. Con este tamaño se podría trabajar de manera correcta sin poner muy en entredicho el tiempo que tarda el módulo de gestión de imágenes.

BÚSQUEDA DE INFORMACIÓN EN LA IMAGEN

Un cambio en la búsqueda de datos en la imagen podría ser otro camino para potenciar este proyecto. Se podrían crear líneas imaginarias que siguieran las líneas que delimitan el carril en lugar de los mismos pixeles que forman la línea. Mediante estas líneas sería posible obtener datos más fiables ya que esas líneas ya representan el patrón de la carretera. El modulo de toma de decisiones tan solo debería comprobar dichas líneas y verificar el estado actual para tomar una decisión.

Figura 49: ejemplo de líneas creadas a partir del análisis de la imagen (Wan, y otros, 2004)

Con esta mejora se podría ahorrar la carga de proceso que tiene el módulo de toma de decisiones pero por el contrario, se cargaría el módulo de gestión de imágenes. Aún así la fiabilidad mejoraría y las decisiones no se verían demasiado afectadas por el posible ruido y molestias de la imagen. Este recurso en imágenes de un mundo real, no virtual, sería el recurso más acertado por que las imágenes reales poseen mucho ruido e impurezas, medios tonos,… En definitiva, gestionar una imagen del mundo real es más complicado que la del mundo virtual que estamos tratando.

GLOSARIO

C#: Microsoft Visual C# 2.0

CCR: Concurrency and Cordination Runtime

ELSIE: Electro-Light-Sensitive Internal-External

IA: Inteligencia artificial

IDE: Interface Development Enviroment

IEEE: The Institute of Electrical and Electronics Engineers

ISO: International Organization for Standardization

JIRA: Japanese Industrial Robotics Association

MIT: Massachusetts Institute Technology

MRDS: Microsoft Robotics Developer Studio

MSVS: Microsoft Visual Studio

MVSL: Microsoft Visual Simulation Environment

MVPL: Microsoft Visual Programming Language

NASREM: NASA Standard Reference Model

NIST: National Institute of Standards and Technology

PUMA: Programmable Universal Machine for Assembly

RIA: Robotics Institute of America

RIC: Representación interna central

RUC: Robochamps Urban Challenge

SRI: Stanford Research Institute

BIBLIOGRAFÍA

Albus, James S., y otros. *NASREM The NASA/NBS Standard Reference Model for Telerobot Control System Architecture.* [En línea] http://www.isd.mel.nist.gov/documents/albus/Loc_143.pdf.

Arkin, Ronald C. 1999. *Behavior-based robotics.* s.l. : MIT Press, 1999. ISBN 0262011654, 9780262011655.

Barber Castaño, Ramón Ignacio. 2000. *Desarrollo de una arquitectura para robots móviles autónomos : aplicación a un sistema de navegación topológica.* [En línea] 14 de Julio de 2000. http://e-archivo.uc3m.es/dspace/handle/10016/577.

Bekey, George A. 2005. *Autonomous Robots: from biological inspiration to implementation and control.* s.l. : MIT Press, 2005. ISBN 0-262-02578-7.

Bernabé Sanchez, Iván. 2007. Proyecto fin de carrera. *Sistema de detección de obstaculos para un robot autónomo móvil.* Escuela politecnica superior, Universidad Carlos III Madrid : s.n., Noviembre de 2007.

Boehm, Barry W. 1988. A Spiral Model of Software Development and Enhacement. [En línea] 1988. http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/spiral.pdf.

Brooks, Rodney A. 1990. *Elephants Don't Play Chess.* [En línea] 1990. http://people.csail.mit.edu/brooks/papers/elephants.pdf.

Canny, J. 1986. A Computational Approach To Edge Detection. *IEEE Trans. Pattern Analysis and Machine Intelligence.* 1986, 8, págs. 679-714.

Chirikjian, Gregory S. y Kyatkin, Alexander B. 2000. *Engineering applications of noncommutative harmonic analysis: with emphasis on rotation and motion groups.* s.l. : CRC Press, 2000. ISBN 0849307481, 9780849307485.

Collado, Juan M., y otros. 2006. *Adaptative Road Lanes Detection and Classification.* Lecture notes in computer science : s.n., 2006. págs. 1151-1162. Vol. 4179.

Errecalde, Marcelo Luis. 2006. *Agentes y sistemas multiagente.* [En línea] Septiembre de 2006. http://www.dirinfo.unsl.edu.ar/~sma/Teorias/teo2ag4.pdf.

Friedenberg, Jay y Silverman, Gordon. 2006. *Cognitive Science: An Introduction to the Study of Mind.* s.l. : SAGE, 2006. ISBN 1412925681, 9781412925686.

Gallegos, Maria Soledad y Gorostegui, Maria Elena. *Procesos cognitivos.* [En línea] http://www.unheval.edu.pe/docente/administrador/subidas/1190494636.pdf.

Ge, Shuzhi Sam y Lewis, Frank L. 2006. *Autonomous Mobile Robots: Sensing, Control, Decision-making, and Applications.* s.l. : CRC Press, 2006. ISBN 0849337488, 9780849337482.

González Marcos, Ana, y otros. 2006. *Técnicas y algoritmos básicos de visión artificial.* s.l. : Servicio de publicaciones, Universidad de La Rioja, 2006. ISBN 846899345X.

Laoz-Beltrá, Rafael. 2004. *Bioinformática: simulación, vida artificial e inteligencia artificial.* s.l. : Ediciones Díaz de Santos, 2004. 8479786450, 9788479786458.

Lope, Javier de. 2006. *Arquitecturas de control de robots moviles.* [En línea] 8 de Noviembre de 2006. http://www.dia.fi.upm.es/~jdlope/slides/arquit.pdf.

McKerrow, Philip John. 1991. *Introduction to Robotics.* s.l. : Addison-Wesley, 1991. ISBN 0-201-18240-8.

Meystel, A. 1991. *Autonomous Mobile Robots: Vehicles with Cognitive Control.* s.l. : World Scientific, 1991. ISBN 9971500892, 9789971500894.

Moriello, Sergio Alejandro. 2005. *Tendencias21.* [En línea] 10 de Octubre de 2005. http://www.tendencias21.net/Los-Robots-Inteligentes-Autonomos-son-la-nueva-generacion_a744.html.

Murphy, Robin. 2000. *Introduction to IA robotics.* s.l. : MIT Press, 2000. ISBN 0262133830, 9780262133838.

Nalwa, Vishvjit S. 1993. *A Guided Tour of Computer Vision.* s.l. : Addison-Wesley, 1993. ISBN 0201548534.

Nehmzow, Ulrich. 2003. *Mobile robotics: a practical introduction.* s.l. : Springer, 2003. ISBN 1852337265, 9781852337261.

Pavón Mestras, Juan. 2006. *Agentes Inteligentes.* [En línea] 2006. http://www.fdi.ucm.es/profesor/jpavon/doctorado/arquitecturas.pdf.

Piattini Velthius, Mario Gerardo, y otros. 2004. *Análisis y diseño detallado de aplicaciones informátcas de gestión.* s.l. : RA-MA, 2004. 847897587X, 9788478975877.

Pratt, William K. 1991. *Digital image processing.* s.l. : Wiley-Interscience, 1991. ISBN 0-471-85766-1.

Sensor-based navigation of a mobile robot in an indoor environment. **Maaref, H. y Barret, C. 2002.** 38, s.l. : Elsevier, 2002, Robotics and Autonomous Systems, págs. 1-18.

Siegwart, Roland y Nourbakhsh, Illah R. 2004. *Introduction to Autonomous Mobile Robots.* s.l. : MIT Press, 2004. ISBN 026219502X, 9780262195027.

Sloman, Aaron y Logan, Brian. 1999. *BUILDING COGNITIVELY RICH AGENTS USING THE.* [En línea] 1999. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.39.1273.

Sucar, L. Enrique. 2008. *Introducción a la Robótica.* [En línea] 2008. http://ccc.inaoep.mx/~esucar/Clases-irob/ir5-arq.pdf.

Valverde Rebaza, Jorge. 2007. Detección de bordes mediante el algoritmo de Canny. [En línea] 2007. [Citado el: 09 de Julio de 2009.] http://www.seccperu.org/files/Detecci%C3%B3ndeBordes-Canny.pdf.

Wan, Yue, Khwang Teoh, Eam y Shen, Dinggang. 2004. *Lane detection and tracking using B-Snake.* Image and Vision Computing : Elsevier, 2004. págs. 269-280. Vol. 22.

Xie, Min. 2003. *Fundamentals of Robotics: Linking Perception to Action.* s.l. : World Scientific, 2003. ISBN 9812383352, 9789812383358.

ANEXOS

ANEXO A: CONTENIDO DEL CD

Junto con el documento impreso se anexa un CD que contiene el código fuente y otros datos necesarios para la ejecución del proyecto. En este anexo se explicará de manera detallada cómo ejecutar el proyecto en cualquier ordenador personal que disponga de Microsoft Visual Studio y Microsoft Robotics Developer Studio.

De no disponer de MRDS, se facilita una versión, la misma con la que se ha realizado el proyecto, para poder ejecutar este proyecto.

INSTALACIÓN DE PROGRAMAS NECESARIOS Y DE LOS FICHEROS DEL PROYECTO

Paso 1: instalación del software previo necesario: como se ha comentado en necesario instalar el MSVS, la versión utilizada para el proyecto es la 2008, y MRDS cuya versión es 2008 (CTP July).

Paso 2: instalación del starter kit de RUC: para el buen funcionamiento del proyecto es necesario instalar el starter kit suministrado por RUC. Aunque la página web ha desaparecido, se dispone de una copia descargada, con lo que solo es necesario ejecutar el archivo.

Una vez termine la instalación puede aparecer un error relacionado con el código inicial, algo que se puede no tener en cuenta ya que no se va a realizar nada con ese código inicial, sino que se va a copiar el proyecto encima de dicho código.

Paso 3: copiar proyecto entregado en la ruta indicada: se procede a la instalación del proyecto en sí, para ello es necesario extraer del archivo comprimido "*código.rar*" en la ruta "*C:\Documents and Settings\NOMBRE DE USUARIO\Microsoft Robotics Dev Studio 2008*". Cuando solicite sobrescribir, se acepta la respuesta y se instalará el proyecto en el ordenador.

CONFIGURACIÓN DEL PROYECTO PARA SU CORRECTO FUNCIONAMIENTO

Para que el proyecto pueda funcionar, debido a que es un proyecto que en inicio fue una base estándar para adaptarse a cualquier ordenador, es necesario especificar un par de rutas que son distintas dependiendo del ordenador y usuario que lo esté utilizando dentro de las propiedades del proyecto.

Paso 1: acceder a las propiedades del proyecto: para acceder a las propiedades del proyecto, ya dentro de Visual Studio, se debe hacer doble click sobre el icono *"UrbanChallenge.sln"* en la ruta *"C:\Documents and Settings\NOMBRE DE USUARIO\Microsoft Robotics Dev Studio 2008\RoboChamps\KIA Urban Challenge\UrbanChallengeEntry\c#".*

Una vez dentro de Visual Studio, se debe abrir las propiedades de la solución. En el lado derecho de la pantalla debe haber un cuadro llamado "Explorador de soluciones", de no aparecer este cuadro es necesario abrirlo mediante la barra superior de menús, en el menú ver y la opción "Explorador de soluciones". Una vez se sitúe en el explorador de soluciones, se hace click derecho sobre el icono de *"UrbanChallenge"*, justo debajo de *"Solución 'UrbanChallenge'"*. En el menú que se despliega, se elije propiedades.

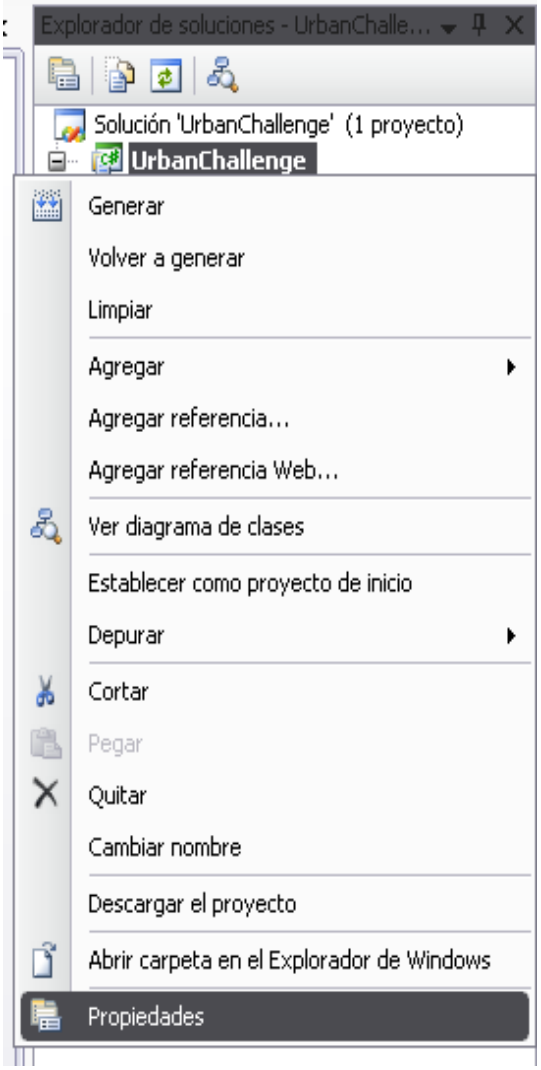


Figura 50: menú desplegable en el explorador de soluciones de MSVS.

Paso 2: una vez abiertas las propiedades del proyecto, se accede a una ventana donde aparecen multitud de parámetros a modificar. Los parámetros que se deben modificar para que el proyecto función correctamente son:

En las opciones a modificar o personalizar, se hace click sobre *"Depurar"*, en la Figura 51 el botón 1. Dentro de la pestaña depurar, es necesario modificar dos líneas. La primera especifica la ruta del ejecutable con el que se va a ejecutar el proyecto. Este proyecto necesita del servicio base que se ejecuta siempre con todo proyecto de robótica dentro de la plataforma MRDS. Para ello es necesario cambiar la ruta de dicho programa ejecutable. Se hace click en el botón 2(véase Figura 51) y elegimos el archivo *"DSShost32.exe"* que se debe encontrar, si se ha realizado una instalación estándar de MRDS, en *"C:\Documents and Settings\NOMBRE DE USUARIO\Microsoft Robotics Dev Studio 2008\bin\"*.

Además de esa ruta, también es necesario modificar la ruta que especifica el directorio de trabajo. Para ello se pulsa el botón 3 (véase Figura 51) y se selecciona el directorio de trabajo que en este caso será *"C:\Documents and Settings\NOMBRE DE USUARIO\Microsoft Robotics Dev Studio 2008\"*.

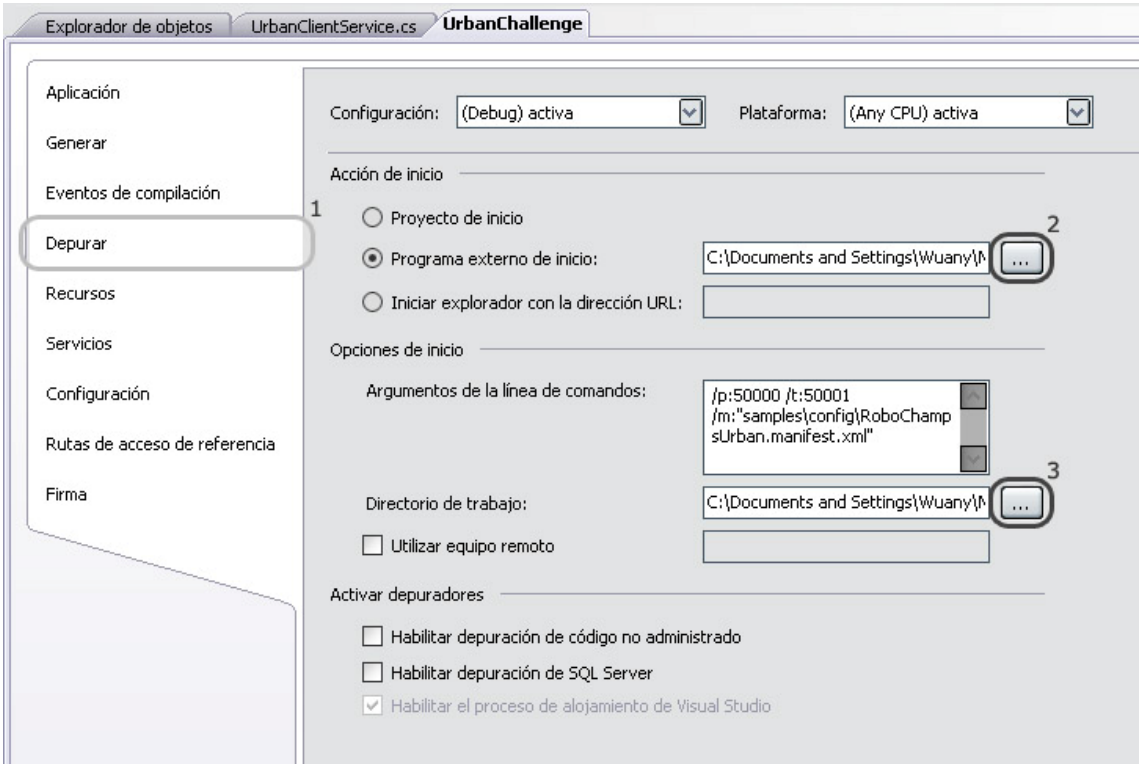
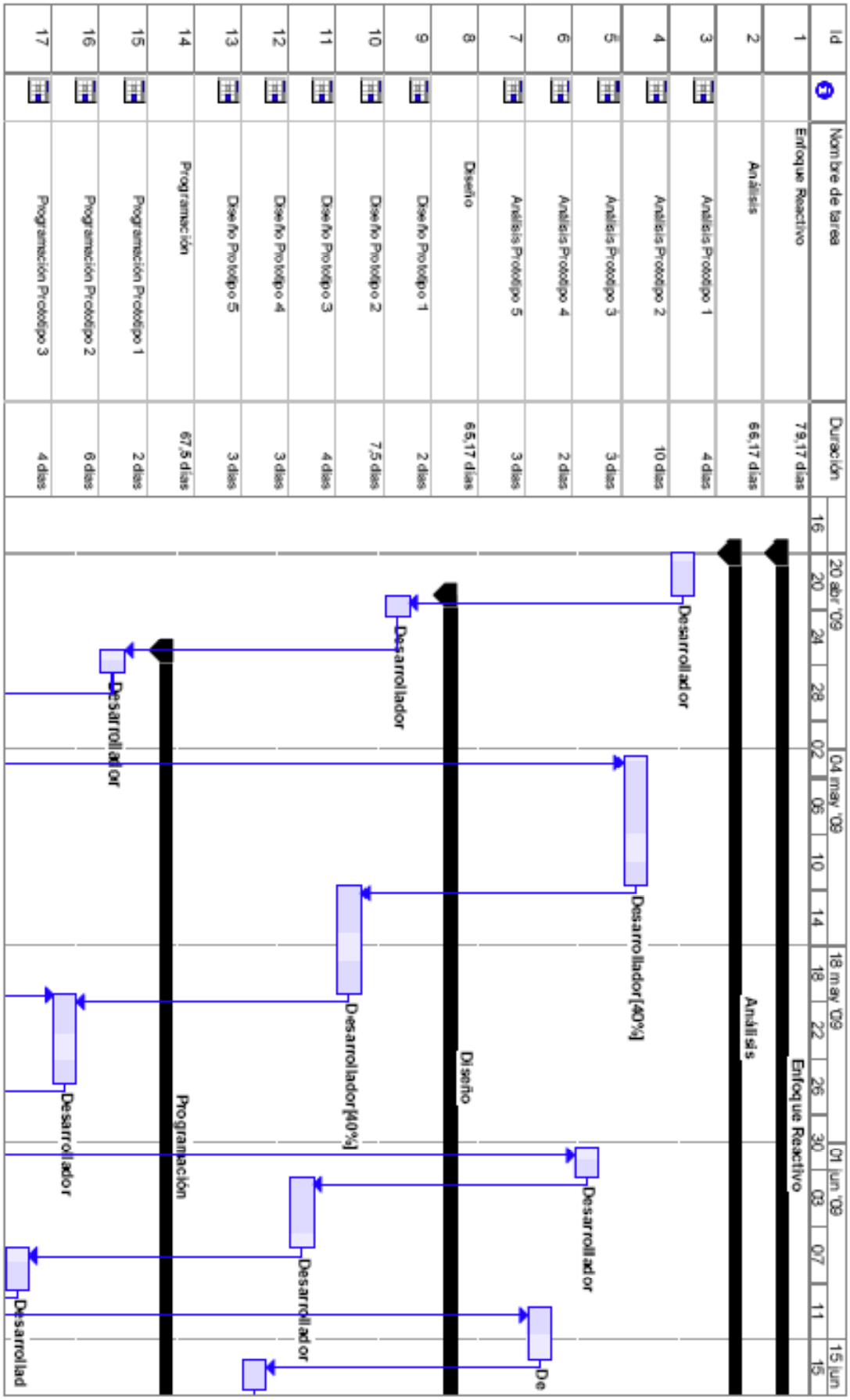


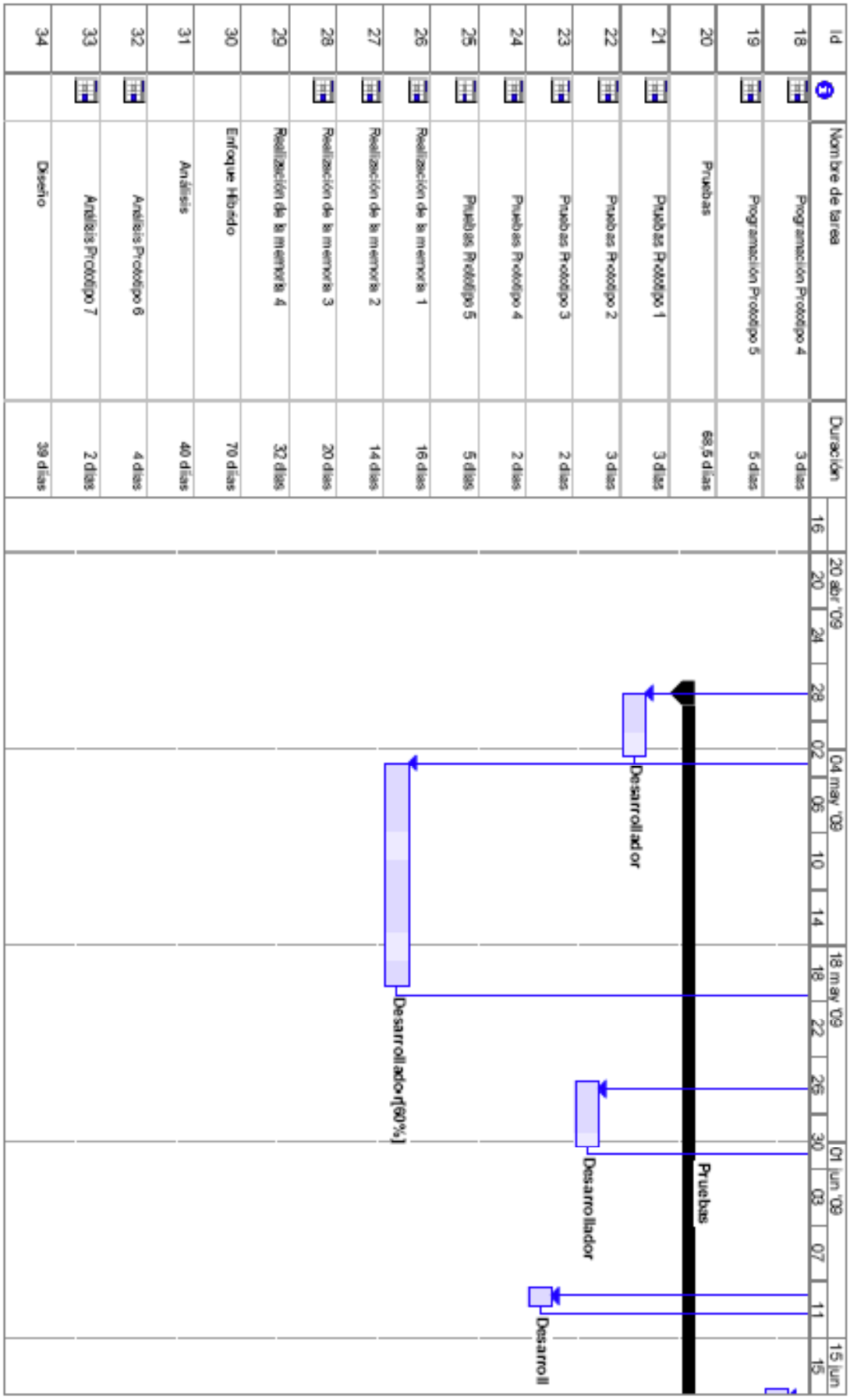
Figura 51: ventana del menú *Depurar* en las propiedades del proyecto.

Una vez realizadas estas operaciones, el proyecto ya estaría en disposición de ejecutarse de manera correcta.

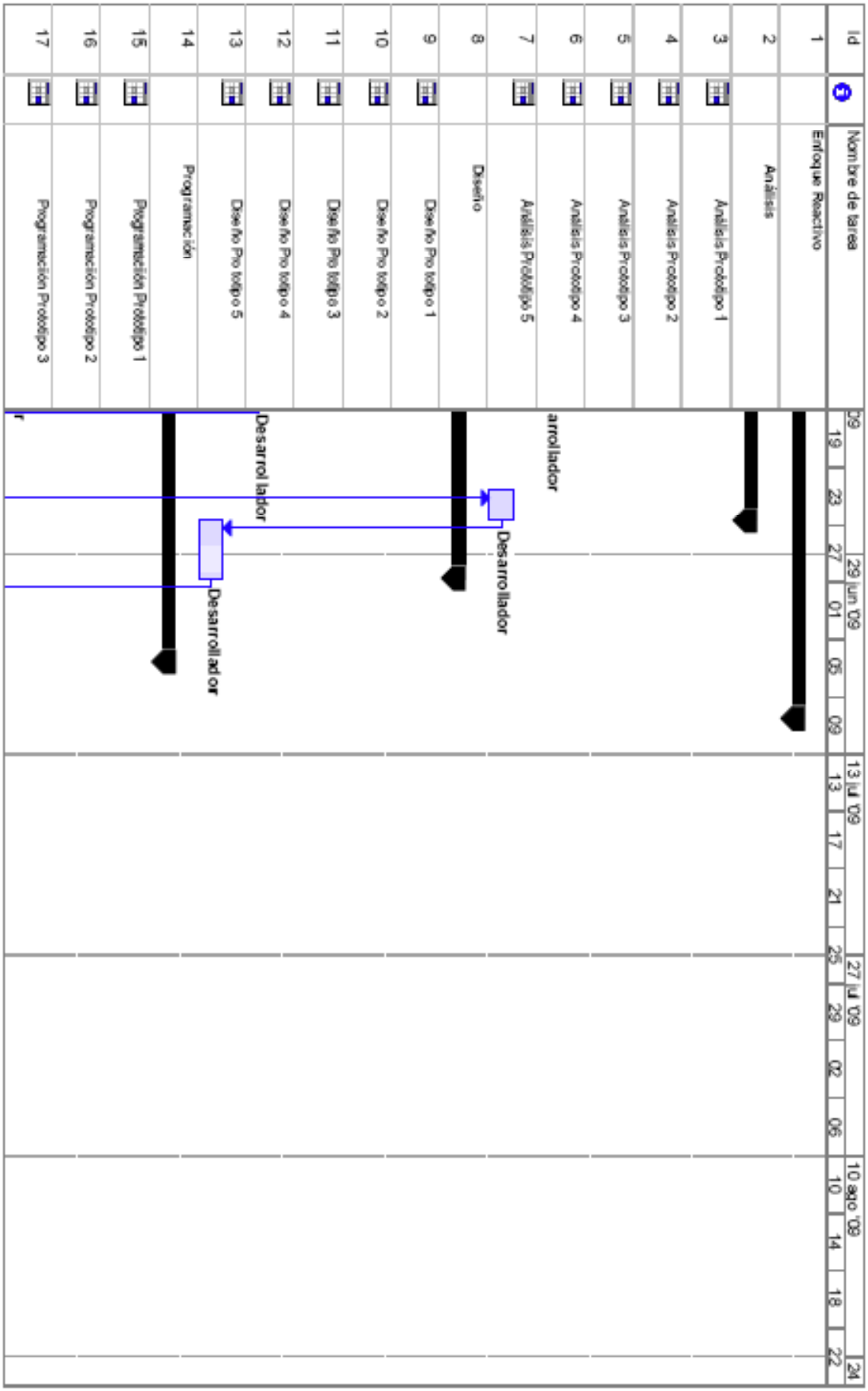
ANEXO B: PLANIFICACIÓN DEL PROYECTO

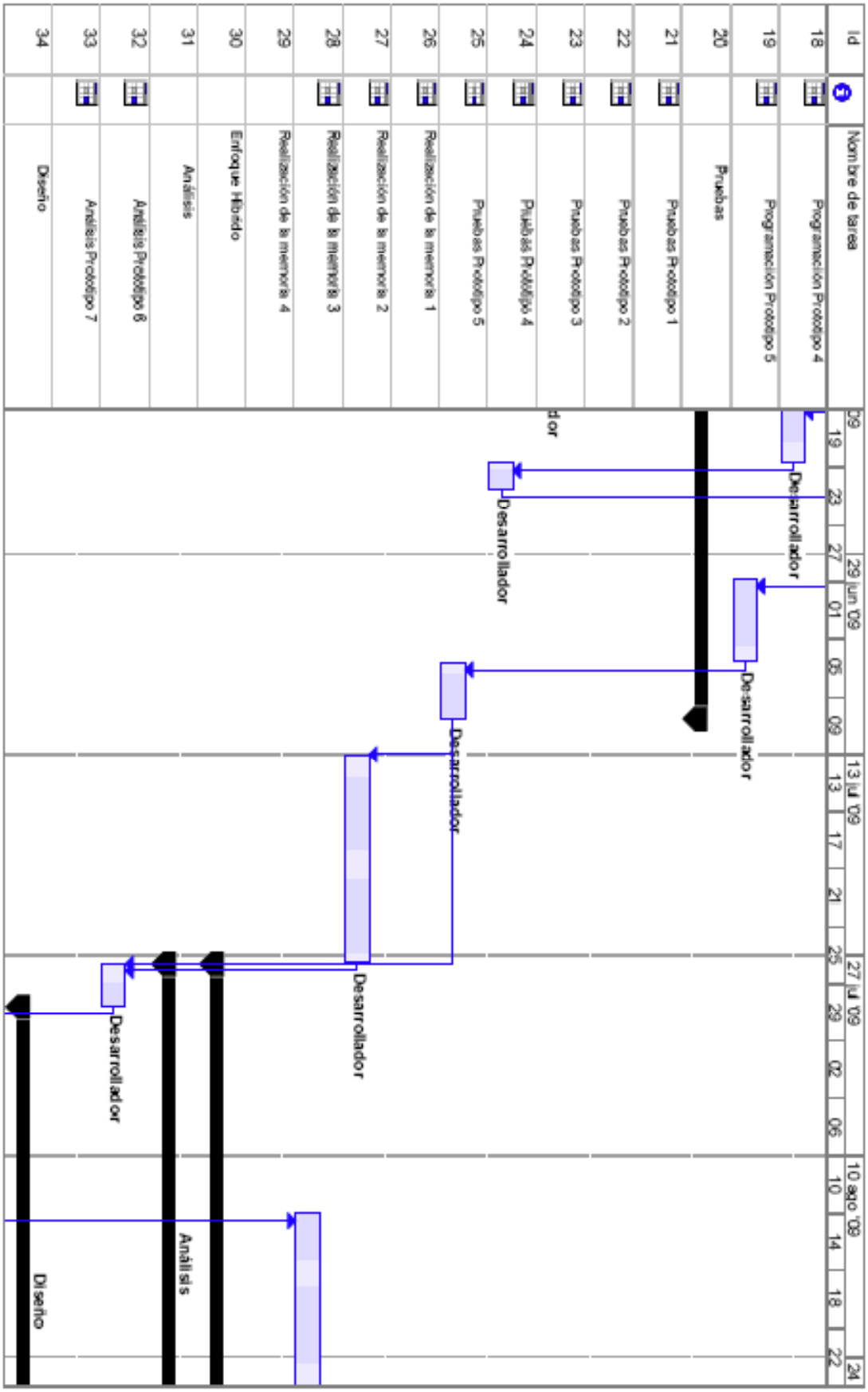
Debido a lo aparatoso que resulta incluir la planificación en el apartado correspondiente de gestión del proyecto, se ha preferido enviar a esta sección dicho apartado. Aún así la legibilidad del mismo es algo complicada debido a que se ha realizado un diagrama de Gantt indicando las tareas realizadas y es algo aparatoso.

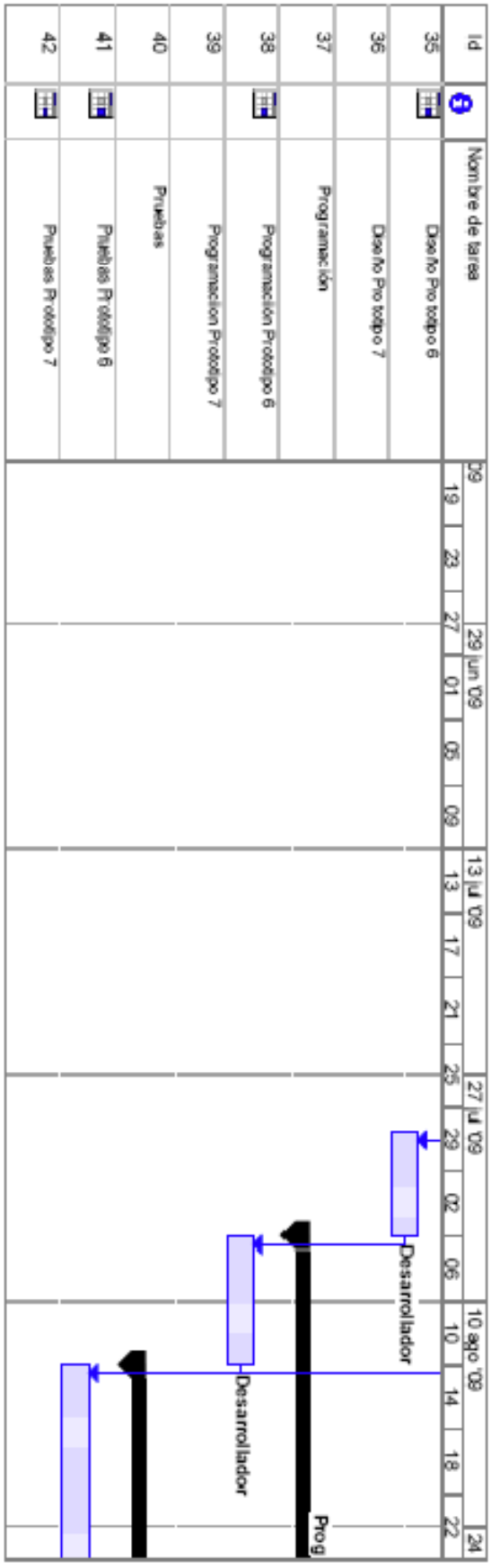

Id
Nombre de tarea
Duración
1
Enfoque Reactivo
79,17 días
2
Análisis
65,17 días
3
Análisis Prototipo 1
4 días
4
Análisis Prototipo 2
10 días
5
Análisis Prototipo 3
3 días
6
Análisis Prototipo 4
2 días
7
Análisis Prototipo 5
3 días
8
Diseño
65,17 días
9
Diseño Prototipo 1
2 días
10
Diseño Prototipo 2
7,5 días
11
Diseño Prototipo 3
4 días
12
Diseño Prototipo 4
3 días
13
Diseño Prototipo 5
3 días
14
Programación
67,5 días
15
Programación Prototipo 1
2 días
16
Programación Prototipo 2
6 días
17
Programación Prototipo 3
4 días
20 abr '09
04 may '09
18 may '09
01 jun '09
15 jun
Desarrollador
Desarrollador[40%]
Desarrollador
Desarrollador
Desarrollador
Desarrollador
Desarrollad
Enfoque Reactivo
Análisis
Diseño
Programación

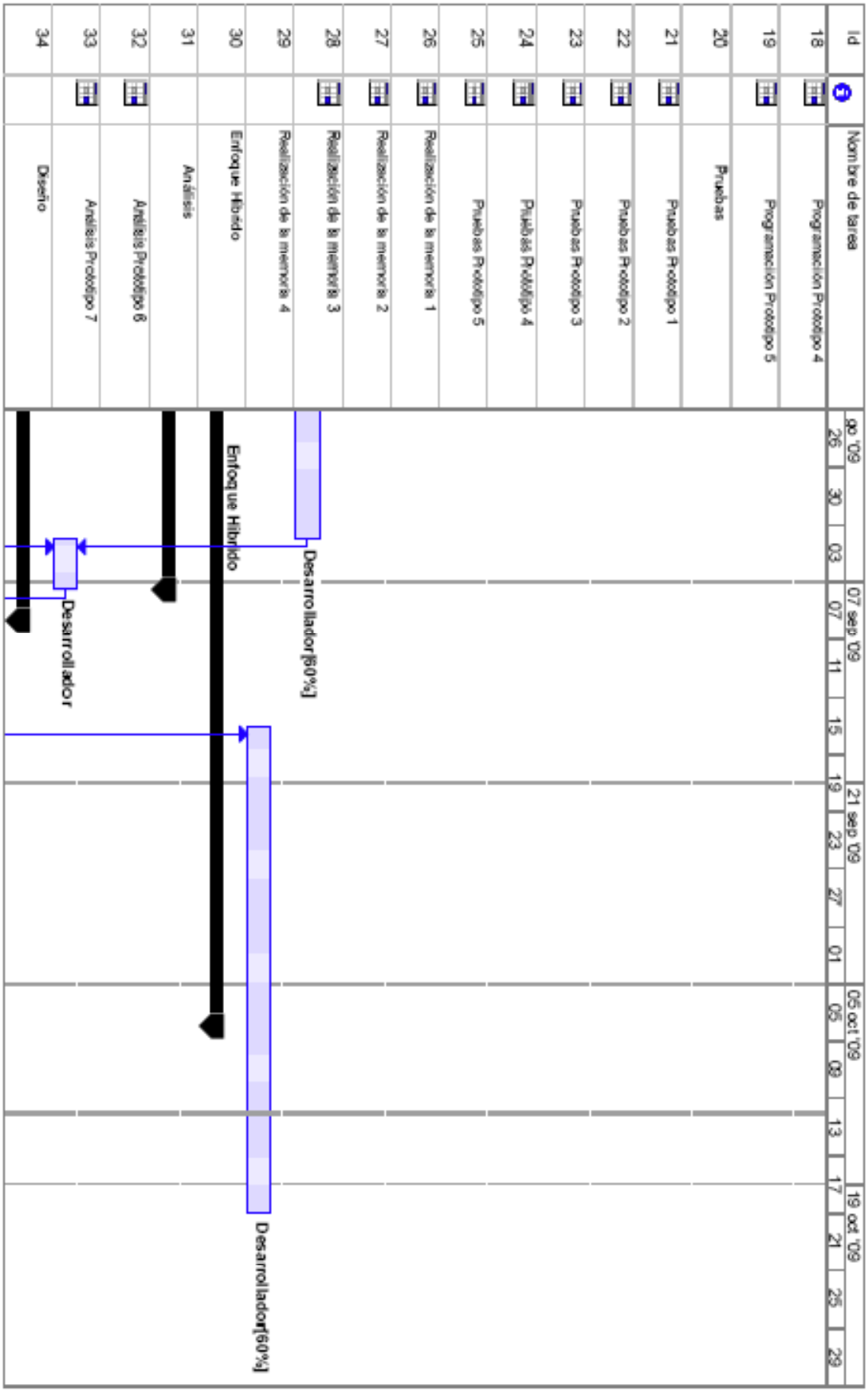

Id
Nombre de tarea
Duración
18 Programación Prototipo 4 3 días
19 Programación Prototipo 5 5 días
20 Pruebas 68,5 días
21 Pruebas Prototipo 1 3 días
22 Pruebas Prototipo 2 3 días
23 Pruebas Prototipo 3 2 días
24 Pruebas Prototipo 4 2 días
25 Pruebas Prototipo 5 5 días
26 Realización de la memoria 1 16 días
27 Realización de la memoria 2 14 días
28 Realización de la memoria 3 20 días
29 Realización de la memoria 4 32 días
30 Enfoque Híbrido 70 días
31 Análisis 40 días
32 Análisis Prototipo 6 4 días
33 Análisis Prototipo 7 2 días
34 Diseño 39 días
20 abr '09
04 may '09
18 may '09
01 jun '09
15 jun
Desarrollador
Desarrollador[60%]
Desarrollador
Pruebas
Desarroll

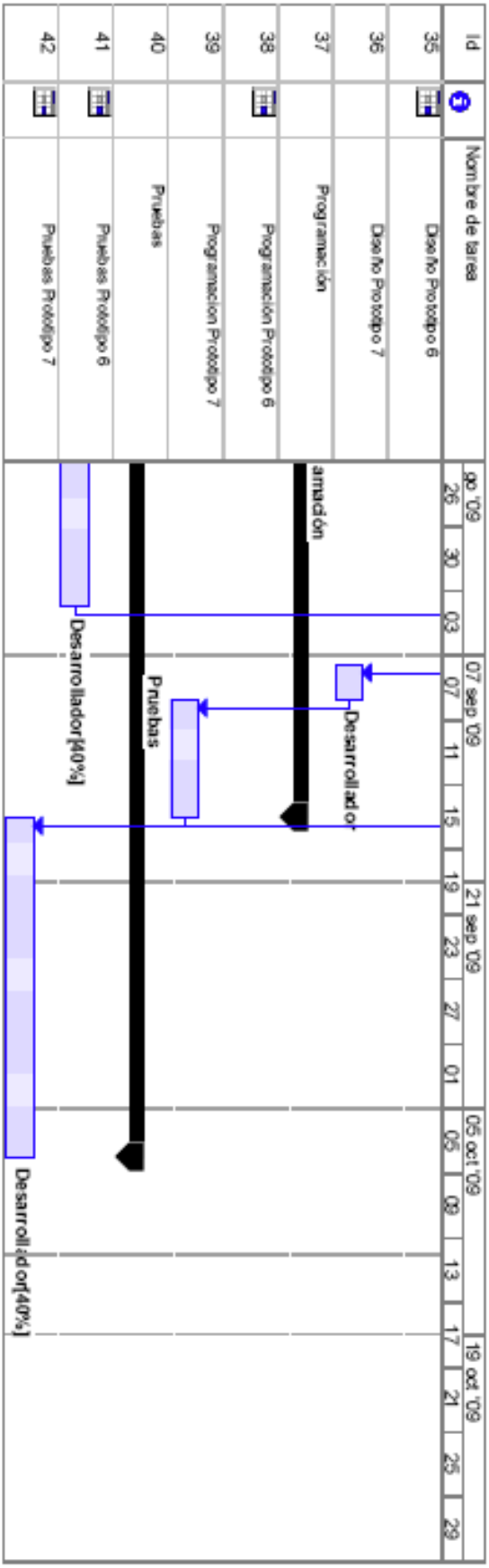
Id	Nombre de tarea	Duración	16	20 abr '09				04 may '09				18 may '09				01 jun '09				15 jun
				20	24	28	02	06	10	14	18	22	26	30	03	07	11	15		
35	Diseño Prototipo 6	6 días																		
36	Diseño Prototipo 7	3 días																		
37	Programación	40 días																		
38	Programación Prototipo 6	8 días																		
39	Programacion Prototipo 7	7 días																		
40	Pruebas	52 días																		
41	Pruebas Prototipo 6	20 días																		
42	Pruebas Prototipo 7	20 días																		


Id
Nombre de tarea
Enfoque Reactivo
Análisis
Análisis Prototipo 1
Análisis Prototipo 2
Análisis Prototipo 3
Análisis Prototipo 4
Análisis Prototipo 5
Diseño
Diseño Prototipo 1
Diseño Prototipo 2
Diseño Prototipo 3
Diseño Prototipo 4
Diseño Prototipo 5
Programación
Programación Prototipo 1
Programación Prototipo 2
Programación Prototipo 3
29 jun '09
13 jul '09
27 jul '09
10 ago '09
Desarrollador
Desarrollador
Desarrollador


Id
Nombre de tarea
Programación Prototipo 4
Programación Prototipo 5
Pruebas
Pruebas Prototipo 1
Pruebas Prototipo 2
Pruebas Prototipo 3
Pruebas Prototipo 4
Pruebas Prototipo 5
Realización de la memoria 1
Realización de la memoria 2
Realización de la memoria 3
Realización de la memoria 4
Enfoque Híbrido
Análisis
Análisis Prototipo 6
Análisis Prototipo 7
Diseño
29 jun '09
13 jul '09
27 jul '09
10 ago '09
Desarrollador
Desarrollador
Desarrollador
Desarrollador
Desarrollador
Análisis
Diseño


Id
Nombre de tarea
35 Diseño Pro totipo 6
36 Diseño Pro totipo 7
37 Programación
38 Programación Prototipo 6
39 Programacion Prototipo 7
40 Pruebas
41 Pruebas Prototipo 6
42 Pruebas Prototipo 7
29 jun '09
13 jul '09
27 jul '09
10 ago '09
Desarrollador
Desarrollador
Prog


Id
Nombre de tarea
Programación Prototipo 4
Programación Prototipo 5
Pruebas
Pruebas Prototipo 1
Pruebas Prototipo 2
Pruebas Prototipo 3
Pruebas Prototipo 4
Pruebas Prototipo 5
Realización de la memoria 1
Realización de la memoria 2
Realización de la memoria 3
Realización de la memoria 4
Enfoque Híbrido
Análisis
Análisis Prototipo 6
Análisis Prototipo 7
Diseño
30 ago '09
07 sep '09
21 sep '09
05 oct '09
19 oct '09
Desarrollador[60%]
Desarrollador
Enfoque Híbrido
Desarrollador[60%]


Id
Nombre de tarea
35 Diseño Prototipo 6
36 Diseño Prototipo 7
37 Programación
38 Programación Prototipo 6
39 Programación Prototipo 7
40 Pruebas
41 Pruebas Prototipo 6
42 Pruebas Prototipo 7
ep '09
26 30 03
07 sep '09
07 11 15
21 sep '09
19 23 27 01
05 oct '09
05 09 13 17
19 oct '09
21 25 29
amación
Desarrollador
Pruebas
Desarrollador[40%]
Desarrollador[40%]